

A Modular Splitting Framework for Saturation Theorem Proving

Ghilain Bergeron, Florent Krasnopol and Sophie Tournet

September 1, 2025

Abstract

We formalize in Isabelle/HOL a framework for splitting [2], a theorem proving technique that extends saturation-based calculi with branching abilities. The framework preserves the completeness of the original calculus. We focus here on the simplest splitting model described in details in the first three sections of "Unifying Splitting" by Gabriel Ebner, Jasmin Blanchette and Sophie Tournet [3] and provide an extension of the ordered resolution calculus with a variant of splitting called Lightweight AVATAR. A paper describing the present formalization has been accepted at ITP'25 [1].

Contents

1	Limsup of Lazy Lists	2
1.1	Library	2
1.2	Infimum	2
1.3	Infimum up-to	3
1.4	Limsup	4
1.5	Limsup up-to	5
2	Disjunctive Consequence Relations	8
3	Extension to Negated Formulas	9
4	Calculi and Derivations	11
5	Annotated Formulas and Consequence Relations	14
6	Lifting Calculi to Add Annotations	16
6.1	Local saturation	26

7	Lifting to Non-ground Calculi	27
7.1	Standard Lifting	27
7.2	Strong Standard Lifting	28
7.3	Lifting with a Family of Tiebreaker Orderings	29
7.4	Lifting with a Family of Redundancy Criteria	32
8	Core splitting calculus	36
8.1	The inference rules	37
8.2	The redundancy criterion	38
8.3	Standard completeness	40
8.4	Strong completeness	41
9	Extensions: Inferences and simplifications	42
9.1	Simplifications	42
9.1.1	The Split Rule	43
9.1.2	The Collect Rule	47
9.1.3	The Trim Rule	48
9.2	Extra Inferences	50
9.2.1	The StrongUnsat Rule	51
9.2.2	The Tauto Rule	52
9.2.3	The Approx Rule	53
9.3	Combining all simplifications and optional inferences	55
9.4	The BinSplit Simplification Rule	56
9.5	Ordered Resolution with a Disjunctive Consequence Relation	58
9.6	Lightweight Avatar without BinSplit	61
9.7	Lightweight Avatar	62

1 Limsup of Lazy Lists

```
theory Lazy-List-Limsup
  imports Coinductive.Coinductive-List
begin
```

1.1 Library

```
lemma less-llength-ltake:  $i < \text{llength } (\text{ltake } k \text{ } Xs) \longleftrightarrow i < k \wedge i < \text{llength } Xs$ 
  <proof>
```

1.2 Infimum

```
definition Inf-llist ::  $\langle 'a \text{ set } llist \Rightarrow 'a \text{ set} \rangle$  where
   $\langle \text{Inf-llist } Xs = (\bigcap i \in \{i. \text{enat } i < \text{llength } Xs\}. \text{lnth } Xs \text{ } i) \rangle$ 
```

```
lemma Inf-llist-subset-lnth:  $\langle \text{enat } i < \text{llength } Xs \Longrightarrow \text{Inf-llist } Xs \subseteq \text{lnth } Xs \text{ } i \rangle$ 
  <proof>
```

lemma *Inf-llist-imp-exists-index*:

assumes $\langle \neg \text{lnull } Xs \rangle$

shows $\langle x \in \text{Inf-llist } Xs \implies \exists i. \text{enat } i < \text{llength } Xs \wedge x \in \text{lnth } Xs \ i \rangle$

$\langle \text{proof} \rangle$

lemma *Inf-llist-imp-all-index*: $\langle x \in \text{Inf-llist } Xs \implies \forall i. \text{enat } i < \text{llength } Xs \longrightarrow x \in \text{lnth } Xs \ i \rangle$

$\langle \text{proof} \rangle$

lemma *all-index-imp-Inf-llist*: $\langle \forall i. \text{enat } i < \text{llength } Xs \longrightarrow x \in \text{lnth } Xs \ i \implies x \in \text{Inf-llist } Xs \rangle$

$\langle \text{proof} \rangle$

lemma *Inf-llist-LNil[simp]*: $\langle \text{Inf-llist } LNil = UNIV \rangle$

$\langle \text{proof} \rangle$

lemma *Inf-llist-LCons[simp]*: $\langle \text{Inf-llist } (LCons \ X \ Xs) = X \cap \text{Inf-llist } Xs \rangle$

$\langle \text{proof} \rangle$

lemma *lhd-subset-Inf-llist*: $\langle \neg \text{lnull } Xs \implies \text{Inf-llist } Xs \subseteq \text{lhd } Xs \rangle$

$\langle \text{proof} \rangle$

1.3 Infimum up-to

definition *Inf-upto-llist* :: $\langle 'a \text{ set } \text{llist} \Rightarrow \text{enat} \Rightarrow 'a \text{ set} \rangle$ **where**

$\langle \text{Inf-upto-llist } Xs \ j = (\bigcap i \in \{i. \text{enat } i < \text{llength } Xs \wedge \text{enat } i \leq j\}. \text{lnth } Xs \ i) \rangle$

lemma *Inf-upto-llist-eq-Inf-llist-ltake*: $\langle \text{Inf-upto-llist } Xs \ j = \text{Inf-llist } (\text{ltake } (\text{eSuc } j) \ Xs) \rangle$

$\langle \text{proof} \rangle$

lemma *Inf-upto-llist-enat-0[simp]*:

$\langle \text{Inf-upto-llist } Xs \ (\text{enat } 0) = (\text{if } \text{lnull } Xs \text{ then } UNIV \text{ else } \text{lhd } Xs) \rangle$

$\langle \text{proof} \rangle$

lemma *Inf-upto-llist-Suc[simp]*:

$\langle \text{Inf-upto-llist } Xs \ (\text{enat } (\text{Suc } j)) =$

$\text{Inf-upto-llist } Xs \ (\text{enat } j) \cap (\text{if } \text{enat } (\text{Suc } j) < \text{llength } Xs \text{ then } \text{lnth } Xs \ (\text{Suc } j)$

$\text{else } UNIV) \rangle$

$\langle \text{proof} \rangle$

lemma *Inf-upto-llist-infinity[simp]*: $\langle \text{Inf-upto-llist } Xs \ \infty = \text{Inf-llist } Xs \rangle$

$\langle \text{proof} \rangle$

lemma *Inf-upto-llist-0[simp]*: $\langle \text{Inf-upto-llist } Xs \ 0 = (\text{if } \text{lnull } Xs \text{ then } UNIV \text{ else } \text{lhd } Xs) \rangle$

$\langle \text{proof} \rangle$

lemma *Inf-upto-llist-eSuc[simp]*:

$\langle \text{Inf-upto-llist } Xs \text{ (eSuc } j) =$
 $\text{ (case } j \text{ of}$
 $\text{ enat } k \Rightarrow \text{Inf-upto-llist } Xs \text{ (enat (Suc } k))$
 $\mid \infty \Rightarrow \text{Inf-llist } Xs) \rangle$
 $\langle \text{proof} \rangle$

lemma *Inf-upto-llist-anti[simp]*: $\langle j \leq k \implies \text{Inf-upto-llist } Xs \ k \subseteq \text{Inf-upto-llist } Xs \ j \rangle$
 $\langle \text{proof} \rangle$

lemma *Inf-llist-subset-Inf-upto-llist*: $\langle \text{Inf-llist } Xs \subseteq \text{Inf-upto-llist } Xs \ j \rangle$
 $\langle \text{proof} \rangle$

lemma *elem-Inf-llist-imp-Inf-upto-llist*:
 $\langle x \in \text{Inf-llist } Xs \implies x \in \text{Inf-upto-llist } Xs \text{ (enat } j) \rangle$
 $\langle \text{proof} \rangle$

lemma *Inf-upto-llist-subset-lnth*: $\langle j < \text{length } Xs \implies \text{Inf-upto-llist } Xs \ j \subseteq \text{lnth } Xs \ j \rangle$
 $\langle \text{proof} \rangle$

lemma *Inf-llist-imp-Inf-upto-llist*:
assumes $\langle X \subseteq \text{Inf-llist } Xs \rangle$
shows $\langle X \subseteq \text{Inf-upto-llist } Xs \text{ (enat } k) \rangle$
 $\langle \text{proof} \rangle$

1.4 Limsup

definition *Limsup-llist* :: $\langle 'a \text{ set llist} \Rightarrow 'a \text{ set} \rangle$ **where**
 $\langle \text{Limsup-llist } Xs =$
 $(\bigcap i \in \{i. \text{enat } i < \text{length } Xs\}. \bigcup j \in \{j. i \leq j \wedge \text{enat } j < \text{length } Xs\}. \text{lnth } Xs \ j) \rangle$

lemma *Limsup-llist-LNil[simp]*: $\langle \text{Limsup-llist } LNil = UNIV \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-llist-LCons*:
 $\langle \text{Limsup-llist } (LCons \ X \ Xs) = (\text{if } lnull \ Xs \text{ then } X \text{ else } \text{Limsup-llist } Xs) \rangle$ (**is** $\langle ?lhs = ?rhs \rangle$)
 $\langle \text{proof} \rangle$

lemma *lfinite-Limsup-llist*: $\langle lfinite \ Xs \implies \text{Limsup-llist } Xs = (\text{if } lnull \ Xs \text{ then } UNIV \text{ else } llast \ Xs) \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-llist-ltl*: $\langle \neg lnull \ (ltl \ Xs) \implies \text{Limsup-llist } Xs = \text{Limsup-llist } (ltl \ Xs) \rangle$
 $\langle \text{proof} \rangle$

lemma *Inf-llist-subset-Limsup-llist*: $\langle \text{Inf-llist } Xs \subseteq \text{Limsup-llist } Xs \rangle$
 $\langle \text{proof} \rangle$

lemma *image-Limsup-llist-subset*: $\langle f ' \text{Limsup-llist } Ns \subseteq \text{Limsup-llist } (\text{lmap } ((\cdot) f) Ns) \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-llist-imp-exists-index*:
assumes $\langle \neg \text{lnull } Xs \rangle$
shows $\langle x \in \text{Limsup-llist } Xs \implies \exists i. \text{enat } i < \text{llength } Xs \wedge x \in \text{lnth } Xs \ i \rangle$
 $\langle \text{proof} \rangle$

lemma *finite-subset-Limsup-llist-imp-exists-index*:
assumes
 $\text{nnil}: \langle \neg \text{lnull } Xs \rangle$ **and**
 $\text{fin}: \langle \text{finite } X \rangle$ **and**
 $\text{in-sup}: \langle X \subseteq \text{Limsup-llist } Xs \rangle$
shows $\langle \forall i. \text{enat } i < \text{llength } Xs \longrightarrow X \subseteq (\bigcup j \in \{j. i \leq j \wedge \text{enat } j < \text{llength } Xs\}. \text{lnth } Xs \ j) \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-llist-lmap-image*:
assumes $f\text{-inj}: \langle \text{inj-on } f \ (\text{Inf-llist } (\text{lmap } g \ xs)) \rangle$
shows $\langle \text{Limsup-llist } (\text{lmap } (\lambda x. f ' g \ x) \ xs) = f ' \text{Limsup-llist } (\text{lmap } g \ xs) \rangle$ **(is**
 $\langle ?lhs = ?rhs \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-llist-lmap-union*:
assumes $\langle \forall x \in \text{lset } xs. \forall y \in \text{lset } xs. g \ x \cap h \ y = \{\} \rangle$
shows $\langle \text{Limsup-llist } (\text{lmap } (\lambda x. g \ x \cup h \ x) \ xs) =$
 $\text{Limsup-llist } (\text{lmap } g \ xs) \cup \text{Limsup-llist } (\text{lmap } h \ xs) \rangle$ **(is** $\langle ?lhs = ?rhs \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-set-filter-commute*:
assumes $\langle \neg \text{lnull } Xs \rangle$
shows $\langle \text{Limsup-llist } (\text{lmap } (\lambda X. \{x \in X. p \ x\}) \ Xs) = \{x \in \text{Limsup-llist } Xs. p \ x\} \rangle$ **(is** $\langle ?lhs = ?rhs \rangle$
 $\langle \text{proof} \rangle$

1.5 Limsup up-to

definition *Limsup-upto-llist* :: $\langle 'a \ \text{set} \ \text{llist} \Rightarrow \text{enat} \Rightarrow 'a \ \text{set} \rangle$ **where**
 $\langle \text{Limsup-upto-llist } Xs \ k =$
 $(\bigcap i \in \{i. \text{enat } i < \text{llength } Xs \wedge \text{enat } i \leq k\}.$
 $\bigcup j \in \{j. i \leq j \wedge \text{enat } j < \text{llength } Xs \wedge \text{enat } j \leq k\}. \text{lnth } Xs \ j) \rangle$

lemma *Limsup-upto-llist-eq-Limsup-llist-ltake*:
 $\langle \text{Limsup-upto-llist } Xs \ j = \text{Limsup-llist } (\text{ltake } (\text{eSuc } j) \ Xs) \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-upto-llist-enat*[simp]:
 $\langle \text{Limsup-upto-llist } Xs \text{ (enat } k) =$
 $(\text{if enat } k < \text{length } Xs \text{ then } \text{lnth } Xs \text{ } k \text{ else if } \text{lnull } Xs \text{ then } UNIV \text{ else } \text{llast } Xs) \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-upto-llist-infinity*[simp]: $\langle \text{Limsup-upto-llist } Xs \infty = \text{Limsup-llist } Xs \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-upto-llist-0*[simp]:
 $\langle \text{Limsup-upto-llist } Xs \text{ } 0 = (\text{if } \text{lnull } Xs \text{ then } UNIV \text{ else } \text{lhs } Xs) \rangle$
 $\langle \text{proof} \rangle$

lemma *Limsup-upto-llist-eSuc*[simp]:
 $\langle \text{Limsup-upto-llist } Xs \text{ (eSuc } j) =$
 $(\text{case } j \text{ of}$
 $\text{enat } k \Rightarrow \text{Limsup-upto-llist } Xs \text{ (enat (Suc } k))$
 $| \infty \Rightarrow \text{Limsup-llist } Xs) \rangle$
 $\langle \text{proof} \rangle$

lemma *Liminf-upto-llist-imp-elem-Limsup-llist*:
assumes $\langle \exists i < \text{length } Xs. \forall j \geq i. j < \text{length } Xs \longrightarrow x \in \text{Limsup-upto-llist } Xs$
 $(\text{enat } j) \rangle$
shows $\langle x \in \text{Limsup-llist } Xs \rangle$
 $\langle \text{proof} \rangle$

end

theory *FSet-Extra*
imports *Main HOL-Library.FSet*
begin

This theory contains some additional lemmas regarding sets and finite sets, which were useful in the process of proving some lemmas in *Modular_Splitting_Calculus.thy* and *Lightweight_Avatar.thy*.

$\langle \text{proof} \rangle$

lemma *finite-because-singleton*: $\langle (\forall C1 \in S. \forall C2 \in S. C1 = C2) \longrightarrow \text{finite } S \rangle$
for S
 $\langle \text{proof} \rangle$

lemma *finite-union-of-finite-is-finite*: $\langle \text{finite } E \implies (\forall D \in E. \text{finite}(\{f \ C \mid C. P \ C \wedge g \ C = D\})) \implies$
 $\text{finite}(\{f \ C \mid C. P \ C \wedge g \ C \in E\}) \rangle$
 $\langle \text{proof} \rangle$

definition *list-of-fset* :: $'a \text{ fset} \Rightarrow 'a \text{ list}$ **where**
 $\langle \text{list-of-fset } A = (\text{SOME } l. \text{fset-of-list } l = A) \rangle$

lemma *fin-set-fset*: $\text{finite } A \implies \exists Af. \text{fset } Af = A \langle \text{proof} \rangle$

lemma *fimage-snd-zip-is-snd* [*simp*]:

$\langle \text{length } x = \text{length } y \implies (\lambda(x, y). y) \mid^! \text{fset-of-list } (\text{zip } x \ y) = \text{fset-of-list } y \rangle$
 $\langle \text{proof} \rangle$

lemma *if-in-ffUnion-then-in-subset*: $\langle x \mid \in \text{ffUnion } A \implies \exists a. a \mid \in A \wedge x \mid \in a \rangle$
 $\langle \text{proof} \rangle$

lemma *fset-ffUnion-subset-iff-all-fsets-subset*: $\langle \text{fset } (\text{ffUnion } A) \subseteq B \longleftrightarrow \text{fBall } A$
 $(\lambda x. \text{fset } x \subseteq B) \rangle$
 $\langle \text{proof} \rangle$

lemma *fBall-fset-of-list-iff-Ball-set*: $\langle \text{fBall } (\text{fset-of-list } A) \ P \longleftrightarrow \text{Ball } (\text{set } A) \ P \rangle$
 $\langle \text{proof} \rangle$

lemma *wf-fsubset*: $\langle \text{wfP } (|\subset|) \rangle$
 $\langle \text{proof} \rangle$

lemma *non-zero-fcard-of-non-empty-set*: $\langle \text{fcard } A > 0 \longleftrightarrow A \neq \{||\} \rangle$
 $\langle \text{proof} \rangle$

lemma *fimage-of-non-fempty-is-non-fempty*: $\langle A \neq \{||\} \implies f \mid^! A \neq \{||\} \rangle$
 $\langle \text{proof} \rangle$

lemma *Union-empty-if-set-empty-or-all-empty*:

$\langle \text{ffUnion } A = \{||\} \implies A = \{||\} \vee \text{fBall } A (\lambda x. x = \{||\}) \rangle$
 $\langle \text{proof} \rangle$

lemma *fBall-fimage-is-fBall*: $\langle \text{fBall } (f \mid^! A) \ P \longleftrightarrow \text{fBall } A (\lambda x. P (f x)) \rangle$
 $\langle \text{proof} \rangle$

lemma *fset-map2*: $\langle v \in \text{fset } A \implies g (f v) \in \text{set } (\text{map } g (\text{map } f (\text{list-of-fset } A))) \rangle$
 $\langle \text{proof} \rangle$

end

theory *Disjunctive-Consequence-Relations*

imports *Saturation-Framework.Calculus*

Propositional-Proof-Systems.Compactness

HOL-Library.Library

HOL-Library.Product-Lexorder

Lazy-List-Limsup

FSet-Extra

begin

2 Disjunctive Consequence Relations

no-notation *Sema.formula-semantics* (**infix** $\models$ 51)

locale *consequence-relation* =

fixes

bot :: 'f **and**

entails :: 'f set $\Rightarrow$ 'f set $\Rightarrow$ bool (**infix** $\models$ 50)

assumes

bot-entails-empty: $\{bot\} \models \{\}$ **and**

entails-reflexive: $\{C\} \models \{C\}$ **and**

entails-subsets: $M' \subseteq M \Rightarrow N' \subseteq N \Rightarrow M' \models N' \Rightarrow M \models N$ **and**

entails-cut: $M \models N \cup \{C\} \Rightarrow M' \cup \{C\} \models N' \Rightarrow M \cup M' \models N \cup N'$ **and**

entails-compactness: $M \models N \Rightarrow \exists M' N'. (M' \subseteq M \wedge N' \subseteq N \wedge \text{finite } M' \wedge \text{finite } N' \wedge M' \models N')$

begin

definition *order-double-subsets* :: ('f set * 'f set) $\Rightarrow$ ('f set * 'f set) $\Rightarrow$ bool

(**infix** $\preceq_s$ 50) **where**

$\langle (\preceq_s) \equiv \lambda C1 C2. \text{fst } C1 \subseteq \text{fst } C2 \wedge \text{snd } C1 \subseteq \text{snd } C2 \rangle$

definition *order-double-subsets-strict* :: ('f set * 'f set) $\Rightarrow$ ('f set * 'f set) $\Rightarrow$ bool

(**infix** $\prec_s$ 50) **where**

$\langle (\prec_s) \equiv \lambda C1 C2. C1 \preceq_s C2 \wedge C1 \neq C2 \rangle$

lemma *trivial-induction-order* : $\langle C1 \subseteq B \wedge C2 \subseteq B' \longrightarrow (C1, C2) \preceq_s (B, B') \rangle$

$\langle \text{proof} \rangle$

lemma *zorn-relation-trans* : $\langle \forall C1 C2 C3. (C1 \preceq_s C2) \longrightarrow (C2 \preceq_s C3) \longrightarrow (C1 \preceq_s C3) \rangle$

$\langle \text{proof} \rangle$

lemma *zorn-strict-relation-trans* :

$\langle \forall (C1 :: 'f set \times 'f set) C2 C3. (C1 \prec_s C2) \longrightarrow (C2 \prec_s C3) \longrightarrow (C1 \prec_s C3) \rangle$

$\langle \text{proof} \rangle$

lemma *zorn-relation-antisym* : $\langle \forall C1 C2. (C1 \preceq_s C2) \longrightarrow (C2 \preceq_s C1) \longrightarrow (C1 = C2) \rangle$

$\langle \text{proof} \rangle$

lemma *entails-supsets* : $\langle \forall M' N'. (M' \supseteq M \wedge N' \supseteq N \wedge M' \cup N' = \text{UNIV}) \longrightarrow M' \models N' \rangle \Rightarrow M \models N$

$\langle \text{proof} \rangle$

lemma *entails-each*: $M \models P \implies \forall C \in M. N \models Q \cup \{C\} \implies \forall D \in P. N \cup \{D\} \models Q \implies N \models Q$
 $\langle proof \rangle$

lemma *entails-bot-to-entails-empty*: $\langle \{\} \models \{bot\} \implies \{\} \models \{\} \rangle$
 $\langle proof \rangle$

abbreviation *equi-entails* :: $'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow bool$ **where**
equi-entails $M \ N \equiv (M \models N \wedge N \models M)$

lemma *entails-cond-reflexive*: $\langle N \neq \{\} \implies N \models N \rangle$
 $\langle proof \rangle$

lemma *entails-empty-reflexive-dangerous*: $\langle \{\} \models \{\} \implies M \models N \rangle$
 $\langle proof \rangle$

definition *entails-conjunctive* :: $'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow bool$ (**infix** $\models \cap$ 50) **where**
 $M \models \cap N \equiv \forall C \in N. M \models \{C\}$

sublocale *Calculus.consequence-relation* $\{bot\}$ ($\models \cap$)
 $\langle proof \rangle$
end

3 Extension to Negated Formulas

datatype $'a \text{ sign} = Pos \ 'a \mid Neg \ 'a$

instance *sign* :: (countable) countable
 $\langle proof \rangle$

fun *neg* :: $\langle 'a \text{ sign} \Rightarrow 'a \text{ sign} \rangle$ **where**
 $\langle neg \ (Pos \ C) = Neg \ C \rangle \mid$
 $\langle neg \ (Neg \ C) = Pos \ C \rangle$

fun *to-V* :: $'a \text{ sign} \Rightarrow 'a$ **where**
 $to-V \ (Pos \ C) = C \mid$
 $to-V \ (Neg \ C) = C$

lemma *neg-neg-A-is-A* [*simp*]: $\langle neg \ (neg \ A) = A \rangle$
 $\langle proof \rangle$

fun *is-Pos* :: $'a \text{ sign} \Rightarrow bool$ **where**
 $is-Pos \ (Pos \ C) = True \mid$
 $is-Pos \ (Neg \ C) = False$

lemma *is-Pos-to-V*: $\langle is-Pos \ C \implies C = Pos \ (to-V \ C) \rangle$

$\langle \text{proof} \rangle$

lemma *is-Neg-to-V*: $\langle \neg \text{is-Pos } C \implies C = \text{Neg } (\text{to-V } C) \rangle$
 $\langle \text{proof} \rangle$

lemma *pos-union-singleton*: $\langle \{D. \text{Pos } D \in N \cup \{\text{Pos } X\}\} = \{D. \text{Pos } D \in N\} \cup \{X\} \rangle$
 $\langle \text{proof} \rangle$

lemma *toV-set[simp]*: $\langle \{\text{to-V } C \mid C. \text{to-V } C \in A\} = A \rangle$
 $\langle \text{proof} \rangle$

lemma *pos-neg-union*: $\langle \{P \ C \mid C. Q \ C \wedge \text{is-Pos } C\} \cup \{P \ C \mid C. Q \ C \wedge \neg \text{is-Pos } C\} = \{P \ C \mid C. Q \ C\} \rangle$
 $\langle \text{proof} \rangle$

context *consequence-relation*

begin

definition *entails-neg* :: $\text{'f sign set} \Rightarrow \text{'f sign set} \Rightarrow \text{bool}$ (**infix** $\models_{\sim}$ 50) **where**
 $\text{entails-neg } M \ N \equiv \{C. \text{Pos } C \in M\} \cup \{C. \text{Neg } C \in N\} \models \{C. \text{Pos } C \in N\} \cup \{C. \text{Neg } C \in M\}$

lemma *swap-neg-in-entails-neg*: $\langle \{\text{neg } A\} \models_{\sim} \{\text{neg } B\} \longleftrightarrow \{B\} \models_{\sim} \{A\} \rangle$
 $\langle \text{proof} \rangle$

lemma *ext-cons-rel*: $\langle \text{consequence-relation } (\text{Pos bot}) \text{ entails-neg} \rangle$
 $\langle \text{proof} \rangle$

interpretation *neg-ext-cons-rel*: *consequence-relation Pos bot entails-neg*
 $\langle \text{proof} \rangle$

lemma *pos-neg-entails-bot*: $\langle \{C\} \cup \{\text{neg } C\} \models_{\sim} \{\text{Pos bot}\} \rangle$
 $\langle \text{proof} \rangle$

lemma *entails-of-entails-iff*:
 $\langle \{C\} \models_{\sim} Cs \implies \text{finite } Cs \implies \text{card } Cs \geq 1 \implies$
 $(\forall C_i \in Cs. \mathcal{M} \cup \{C_i\} \models_{\sim} \{\text{Pos bot}\}) \implies \mathcal{M} \cup \{C\} \models_{\sim} \{\text{Pos bot}\} \rangle$
 $\langle \text{proof} \rangle$

end

end

theory *Calculi-And-Annotations*

imports *Disjunctive-Consequence-Relations*

begin

4 Calculi and Derivations

context

begin

no-notation *Extended.extended.Pinf* ($\langle \infty \rangle$)

typedef *'a infinite-llist* = $\langle \{ l :: 'a \text{ llist. } \text{llength } l = \infty \} \rangle$
morphisms *to-llist Abs-infinite-llist*
 $\langle \text{proof} \rangle$

setup-lifting *type-definition-infinite-llist*

lift-definition *lmap* :: $\langle ('a \Rightarrow 'b) \Rightarrow 'a \text{ infinite-llist} \Rightarrow 'b \text{ infinite-llist} \rangle$ **is** *lmap*
 $\langle \text{proof} \rangle$

lift-definition *llnth* :: $\langle 'a \text{ infinite-llist} \Rightarrow \text{nat} \Rightarrow 'a \rangle$ **is** *lnth* $\langle \text{proof} \rangle$

lift-definition *Liminf-infinite-llist* :: $\langle 'a \text{ set infinite-llist} \Rightarrow 'a \text{ set} \rangle$ **is** *Liminf-llist*
 $\langle \text{proof} \rangle$

lift-definition *Limsup-infinite-llist* :: $\langle 'a \text{ set infinite-llist} \Rightarrow 'a \text{ set} \rangle$ **is** *Limsup-llist*
 $\langle \text{proof} \rangle$

lift-definition *Sup-infinite-llist* :: $\langle 'a \text{ set infinite-llist} \Rightarrow 'a \text{ set} \rangle$ **is** *Sup-llist* $\langle \text{proof} \rangle$

lift-definition *llhd* :: $\langle 'a \text{ infinite-llist} \Rightarrow 'a \rangle$ **is** *lhd* $\langle \text{proof} \rangle$

lemma *llength-of-to-llist-is-infinite*: $\langle \text{llength } (\text{to-llist } L) = \infty \rangle$
 $\langle \text{proof} \rangle$

end

locale *sound-inference-system* =

inference-system *Inf* + *sound-cons*: *consequence-relation* *bot* *entails-sound*
for

Inf :: *'f inference set* **and**

bot :: *'f* **and**

entails-sound :: *'f set* $\Rightarrow$ *'f set* $\Rightarrow$ *bool* (**infix** $\models_s$ 50)

+ **assumes**

sound: $\iota \in \text{Inf} \Longrightarrow \text{set } (\text{prems-of } \iota) \models_s \{\text{concl-of } \iota\}$

no-notation *IArray.sub* (**infixl** !! 100)

definition *is-derivation* :: $('f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow \text{bool}) \Rightarrow ('f \text{ set infinite-llist}) \Rightarrow \text{bool}$
where

is-derivation *R* *Ns* $\equiv \forall i. R (\text{llnth } Ns \ i) (\text{llnth } Ns \ (\text{Suc } i))$

definition *terminates* :: *'f set infinite-llist* $\Rightarrow$ *bool* **where**

$terminates\ Ns \equiv \exists i. \forall j > i. llnth\ Ns\ j = llnth\ Ns\ i$

abbreviation $\langle lim-inf \equiv Liminf-infinite-llist \rangle$

abbreviation $limit :: 'f\ set\ infinite-llist \Rightarrow 'f\ set$ **where** $limit\ Ns \equiv lim-inf\ Ns$

abbreviation $lim-sup :: \langle 'f\ set\ infinite-llist \Rightarrow 'f\ set \rangle$ **where**
 $\langle lim-sup\ Ns \equiv Limsup-infinite-llist\ Ns \rangle$

locale $calculus = inference-system\ Inf + consequence-relation\ bot\ entails$
for

$bot :: 'f$ **and**

$Inf :: \langle 'f\ inference\ set \rangle$ **and**

$entails :: 'f\ set \Rightarrow 'f\ set \Rightarrow bool$ (**infix** $\models 50$)

+ fixes

$Red_I :: 'f\ set \Rightarrow 'f\ inference\ set$ **and**

$Red_F :: 'f\ set \Rightarrow 'f\ set$

assumes

$Red-I-to-Inf: Red_I\ N \subseteq Inf$ **and**

$Red-F-Bot: N \models \{bot\} \Longrightarrow N - Red_F\ N \models \{bot\}$ **and**

$Red-F-of-subset: N \subseteq N' \Longrightarrow Red_F\ N \subseteq Red_F\ N'$ **and**

$Red-I-of-subset: N \subseteq N' \Longrightarrow Red_I\ N \subseteq Red_I\ N'$ **and**

$Red-F-of-Red-F-subset: N' \subseteq Red_F\ N \Longrightarrow Red_F\ N \subseteq Red_F\ (N - N')$ **and**

$Red-I-of-Red-F-subset: N' \subseteq Red_F\ N \Longrightarrow Red_I\ N \subseteq Red_I\ (N - N')$ **and**

$Red-I-of-Inf-to-N: \iota \in Inf \Longrightarrow concl-of\ \iota \in N \Longrightarrow \iota \in Red_I\ N$

begin

definition $saturated :: 'f\ set \Rightarrow bool$ **where**

$saturated\ N \longleftrightarrow Inf-from\ N \subseteq Red_I\ N$

definition $Red_I-strict :: 'f\ set \Rightarrow 'f\ inference\ set$ **where**

$Red_I-strict\ N = \{\iota. \iota \in Red_I\ N \vee (\iota \in Inf \wedge bot \in N)\}$

definition $Red_F-strict :: 'f\ set \Rightarrow 'f\ set$ **where**

$Red_F-strict\ N = \{C. C \in Red_F\ N \vee (bot \in N \wedge C \neq bot)\}$

lemma $strict-calc-if-nobot:$

$\forall N. bot \notin Red_F\ N \Longrightarrow calculus\ bot\ Inf\ entails\ Red_I-strict\ Red_F-strict$
 $\langle proof \rangle$

definition $weakly-fair :: 'f\ set\ infinite-llist \Rightarrow bool$ **where**

$\langle weakly-fair\ Ns \equiv Inf-from\ (Liminf-infinite-llist\ Ns) \subseteq Sup-infinite-llist\ (llmap\ Red_I\ Ns) \rangle$

abbreviation $fair :: 'f\ set\ infinite-llist \Rightarrow bool$ **where** $fair\ N \equiv weakly-fair\ N$

definition $derive :: 'f\ set \Rightarrow 'f\ set \Rightarrow bool$ (**infix** $\triangleright 50$) **where**

$M \triangleright N \equiv (M - N \subseteq Red_F\ N)$

lemma *derive-refl*: $M \triangleright M$ $\langle proof \rangle$

lemma *deriv-red-in*: $\langle M \triangleright N \implies Red_F M \subseteq N \cup Red_F N \rangle$
 $\langle proof \rangle$

lemma *derive-trans*: $M \triangleright N \implies N \triangleright N' \implies M \triangleright N'$
 $\langle proof \rangle$

end

locale *sound-calculus* = *sound-inference-system* *Inf* *bot* *entails-sound* +
consequence-relation *bot* *entails*

for

bot :: 'f **and**

Inf :: <'f inference set> **and**

entails :: 'f set $\Rightarrow$ 'f set $\Rightarrow$ bool (**infix** $\models$ 50) **and**

entails-sound :: 'f set $\Rightarrow$ 'f set $\Rightarrow$ bool (**infix** $\models_s$ 50)

+ **fixes**

Red_I :: 'f set $\Rightarrow$ 'f inference set **and**

Red_F :: 'f set $\Rightarrow$ 'f set

assumes

Red-I-to-Inf: $Red_I N \subseteq Inf$ **and**

Red-F-Bot: $N \models \{bot\} \implies N - Red_F N \models \{bot\}$ **and**

Red-F-of-subset: $N \subseteq N' \implies Red_F N \subseteq Red_F N'$ **and**

Red-I-of-subset: $N \subseteq N' \implies Red_I N \subseteq Red_I N'$ **and**

Red-F-of-Red-F-subset: $N' \subseteq Red_F N \implies Red_F N \subseteq Red_F (N - N')$ **and**

Red-I-of-Red-F-subset: $N' \subseteq Red_F N \implies Red_I N \subseteq Red_I (N - N')$ **and**

Red-I-of-Inf-to-N: $\iota \in Inf \implies concl-of \iota \in N \implies \iota \in Red_I N$

begin

sublocale *calculus* *bot* *Inf* *entails*

$\langle proof \rangle$

end

locale *statically-complete-calculus* = *calculus* +

assumes *statically-complete*: $saturated N \implies N \models \{bot\} \implies bot \in N$

begin

lemma *inf-from-sub*: $M \subseteq N \implies Inf-from M \subseteq Inf-from N$
 $\langle proof \rangle$

lemma *nobot-in-Red*: $\langle bot \notin Red_F N \rangle$
 $\langle proof \rangle$

interpretation *strict-calculus*:
statically-complete-calculus bot Inf entails Red_I-strict Red_F-strict
 $\langle \text{proof} \rangle$

end

locale *dynamically-complete-calculus* = *calculus* +
assumes *dynamically-complete*:
 $\langle \text{is-derivation } (\triangleright) \text{ } Ns \implies \text{fair } Ns \implies \text{llhd } Ns \models \{\text{bot}\} \implies \exists i. \text{bot} \in \text{llnth } Ns \rangle$
i

5 Annotated Formulas and Consequence Relations

datatype (*f*, *'v::countable*) *AF* = *Pair* (*F-of*: *'f*) (*A-of*: *'v sign fset*)

definition *is-interpretation* :: *'v sign set* $\Rightarrow$ *bool* **where**
 $\langle \text{is-interpretation } J = (\forall v1 \in J. (\forall v2 \in J. (\text{to-}V \text{ } v1 = \text{to-}V \text{ } v2 \longrightarrow v1 = v2))) \rangle$

typedef *'v propositional-interpretation* = $\{J :: 'v \text{ sign set. is-interpretation } J\}$
 $\langle \text{proof} \rangle$

abbreviation *interp-of* $\equiv$ *Abs-propositional-interpretation*

abbreviation *strip* $\equiv$ *Rep-propositional-interpretation*

setup-lifting *type-definition-propositional-interpretation*

lift-definition *belong-to* :: *'v sign* $\Rightarrow$ *'v propositional-interpretation* $\Rightarrow$ *bool* (**infix** $\in_J$ 90)

is ($\in$)::(*'v sign* $\Rightarrow$ *'v sign set* $\Rightarrow$ *bool*) $\langle \text{proof} \rangle$

definition *total* :: *'v propositional-interpretation* $\Rightarrow$ *bool* **where**

$\langle \text{total } J \equiv (\forall v. (\exists v_J. v_J \in J \wedge \text{to-}V \text{ } v_J = v)) \rangle$

typedef *'v total-interpretation* = $\{J :: 'v \text{ propositional-interpretation. total } J\}$
 $\langle \text{proof} \rangle$

abbreviation *total-interp-of* $\equiv$ ($\lambda x. \text{Abs-total-interpretation } (\text{interp-of } x)$)

abbreviation *total-strip* $\equiv$ ($\lambda x. \text{strip } (\text{Rep-total-interpretation } x)$)

lemma *neg-notin-total-strip* [*simp*]: $\langle (\text{neg } a \notin \text{total-strip } J) = (a \in \text{total-strip } J) \rangle$
 $\langle \text{proof} \rangle$

lemma *neg-in-total-strip* [*simp*]: $\langle (\text{neg } a \in \text{total-strip } J) = (a \notin \text{total-strip } J) \rangle$
 $\langle \text{proof} \rangle$

setup-lifting *type-definition-total-interpretation*

lift-definition *belong-to-total* :: *'v sign* $\Rightarrow$ *'v total-interpretation* $\Rightarrow$ *bool* (**infix** $\in_t$

90)

is $(\in_J)::('v \text{ sign} \Rightarrow 'v \text{ propositional-interpretation} \Rightarrow \text{bool}) \langle \text{proof} \rangle$

lemma *in-total-to-strip* [simp]: $\langle a \in_t J \longleftrightarrow a \in \text{total-strip } J \rangle$
 $\langle \text{proof} \rangle$

lemma *neg-prop-interp*: $\langle (v::'v \text{ sign}) \in_J J \Longrightarrow \neg ((\text{neg } v) \in_J J) \rangle$
 $\langle \text{proof} \rangle$

lemma *neg-total-interp*: $\langle (v::'v \text{ sign}) \in_t J \Longrightarrow \neg ((\text{neg } v) \in_t J) \rangle$
 $\langle \text{proof} \rangle$

lemma *neg-notin-total-interp*: $\langle \neg (v \in_t J) \Longrightarrow ((\text{neg } v) \in_t J) \rangle$
 $\langle \text{proof} \rangle$

definition *to-AF* :: $'f \Rightarrow ('f, 'v::\text{countable}) \text{ AF}$ **where**
 $\langle \text{to-AF } C = \text{Pair } C \{||\} \rangle$

lemma *F-of-to-AF*: $\langle F\text{-of } (\text{to-AF } C) = C \rangle$
 $\langle \text{proof} \rangle$

lemma *A-of-to-AF*: $\langle A\text{-of } (\text{to-AF } C) = \{||\} \rangle$
 $\langle \text{proof} \rangle$

lemma *F-of-circ-to-AF-is-id* [simp]: $\langle F\text{-of} \circ \text{to-AF} = \text{id} \rangle$
 $\langle \text{proof} \rangle$

lemma *A-of-circ-to-AF-is-empty-set* [simp]: $\langle A\text{-of} \circ \text{to-AF} = (\lambda _. \{||\}) \rangle$
 $\langle \text{proof} \rangle$

lemma *F-of-propositional-clauses* [simp]:
 $\langle (\forall x \in \text{set } \mathcal{N}. F\text{-of } x = \text{bot}) \Longrightarrow \text{map } F\text{-of } \mathcal{N} = \text{map } (\lambda _. \text{bot}) \mathcal{N} \rangle$
 $\langle \text{proof} \rangle$

lemma *F-of-Pair* [simp]: $\langle F\text{-of} \circ (\lambda(x, y). \text{AF.Pair } x y) = (\lambda(x, y). x) \rangle$
 $\langle \text{proof} \rangle$

lemma *A-of-Pair* [simp]: $\langle A\text{-of} \circ (\lambda(x, y). \text{AF.Pair } x y) = (\lambda(x, y). y) \rangle$
 $\langle \text{proof} \rangle$

lemma *map-A-of-map2-Pair*: $\langle \text{length } A = \text{length } B \Longrightarrow \text{map } A\text{-of } (\text{map2 } \text{AF.Pair } A B) = B \rangle$
 $\langle \text{proof} \rangle$

definition *Neg-set* :: $'v \text{ sign set} \Rightarrow 'v \text{ sign set}$ ($\sim$ - 55) **where**
 $\langle \sim V \equiv \{\text{neg } v \mid v. v \in V\} \rangle$

definition *F-of-Inf* :: $((f, 'v::\text{countable}) \text{ AF}) \text{ inference} \Rightarrow 'f \text{ inference}$ **where**
 $\langle F\text{-of-Inf } \iota_{AF} = (\text{Infer } (\text{map } F\text{-of } (\text{prems-of } \iota_{AF})) (F\text{-of } (\text{concl-of } \iota_{AF}))) \rangle$

6 Lifting Calculi to Add Annotations

locale *calculus-with-annotated-consrel* = *sound-calculus bot Inf entails entails-sound*
Red_I Red_F

for
bot :: '*f* **and**
Inf :: ⟨'*f* inference set⟩ **and**
entails :: '*f* set ⇒ '*f* set ⇒ bool (**infix** $\models$ 50) **and**
entails-sound :: '*f* set ⇒ '*f* set ⇒ bool (**infix** $\models_s$ 50) **and**
Red_I :: '*f* set ⇒ '*f* inference set **and**
Red_F :: '*f* set ⇒ '*f* set
+ fixes
fml :: ⟨'*v* :: countable ⇒ '*f*⟩ **and**
asn :: ⟨'*f* sign ⇒ '*v* sign set⟩
assumes
fml-entails-C: ⟨ $\forall a \in \text{asn } C. \text{sound-cons.entails-neg } \{\text{map-sign fml } a\} \{C\}$ ⟩
and
C-entails-fml: ⟨ $\forall a \in \text{asn } C. \text{sound-cons.entails-neg } \{C\} \{\text{map-sign fml } a\}$ ⟩
and
asn-not-empty: ⟨ $\text{asn } C \neq \{\}$ ⟩
begin

notation *sound-cons.entails-neg* (**infix** $\langle \models_{s\sim} \rangle$ 50)

lemma *equi-entails-if-a-in-asns*: ⟨ $a \in \text{asn } C \implies a \in \text{asn } D \implies \{C\} \models_{s\sim} \{D\} \wedge \{D\} \models_{s\sim} \{C\}$ ⟩
 ⟨*proof*⟩

lemma *equi-entails-if-neg-a-in-asn*:
 ⟨ $a \in \text{asn } C \implies \text{neg } a \in \text{asn } D \implies \{C\} \models_{s\sim} \{\text{neg } D\} \wedge \{\text{neg } D\} \models_{s\sim} \{C\}$ ⟩
 ⟨*proof*⟩

definition *ιF -of* :: ('*f*, '*v*) *AF inference* ⇒ '*f* inference **where**
 ⟨ *ιF -of* ι = *Infer* (*List.map* *F-of* (*prems-of* ι)) (*F-of* (*concl-of* ι))⟩

definition *propositional-projection* :: ('*f*, '*v*) *AF set* ⇒ ('*f*, '*v*) *AF set* (*proj_⊥*)
where
 ⟨*proj_⊥* $\mathcal{N}$ = {*C*. *C* ∈ $\mathcal{N}$ ∧ *F-of* *C* = *bot*}⟩

lemma *prop-proj-in*: ⟨*proj_⊥* $\mathcal{N} \subseteq \mathcal{N}$ ⟩
 ⟨*proof*⟩

definition *enabled* :: ('*f*, '*v*) *AF* ⇒ '*v* total-interpretation ⇒ bool **where**
enabled *C* *J* ≡ *fset* (*A-of* *C*) ⊆ (*total-strip* *J*)

lemma *subformula-of-enabled-formula-is-enabled*: ⟨*A-of* *C* $\sqsubset$ *A-of* *C'* ⇒ *enabled* *C'* *J* ⇒ *enabled* *C* *J*⟩
 ⟨*proof*⟩

lemma *enabled-iff*: $\langle A\text{-of } C = A\text{-of } C' \implies \text{enabled } C \ J \longleftrightarrow \text{enabled } C' \ J \rangle$
 $\langle \text{proof} \rangle$

definition *enabled-set* :: $(f, v) \text{ AF set} \Rightarrow v \text{ total-interpretation} \Rightarrow \text{bool}$ **where**
 $\langle \text{enabled-set } \mathcal{N} \ J = (\forall C \in \mathcal{N}. \text{enabled } C \ J) \rangle$

lemma *enabled-set-singleton [simp]*: $\langle \text{enabled-set } \{C\} \ J \longleftrightarrow \text{enabled } C \ J \rangle$
 $\langle \text{proof} \rangle$

definition *enabled-inf* :: $(f, v) \text{ AF inference} \Rightarrow v \text{ total-interpretation} \Rightarrow \text{bool}$
where
 $\langle \text{enabled-inf } \iota \ J = (\forall C \in \text{set } (\text{prems-of } \iota). \text{enabled } C \ J) \rangle$

definition *enabled-projection* :: $(f, v) \text{ AF set} \Rightarrow v \text{ total-interpretation} \Rightarrow f \text{ set}$
 $(\text{infix } \text{proj}_J \ 60)$ **where**
 $\langle \mathcal{N} \ \text{proj}_J \ J = \{F\text{-of } C \mid C. C \in \mathcal{N} \wedge \text{enabled } C \ J\} \rangle$

lemma *Union-of-enabled-projection-is-enabled-projection*: $\langle (\bigcup C \in \mathcal{N}. \{C\} \ \text{proj}_J \ J) = \mathcal{N} \ \text{proj}_J \ J \rangle$
 $\langle \text{proof} \rangle$

lemma *projection-of-enabled-subset*:
 $\langle \text{fset } B \subseteq \text{total-strip } J \implies \{AF.Pair \ C \ (A \mid \cup \mid B)\} \ \text{proj}_J \ J = \{AF.Pair \ C \ A\} \ \text{proj}_J \ J \rangle$
 $\langle \text{proof} \rangle$

lemma *Un-of-enabled-projection-is-enabled-projection-of-Un*:
 $\langle (\bigcup x. P \ x) \ \text{proj}_J \ J = (\bigcup x. P \ x \ \text{proj}_J \ J) \rangle$
 $\langle \text{proof} \rangle$

lemma *enabled-projection-of-Int-is-Int-of-enabled-projection*:
 $\langle x \in (\bigcap S) \ \text{proj}_J \ J \implies x \in \bigcap \{x \ \text{proj}_J \ J \mid x. x \in S\} \rangle$
 $\langle \text{proof} \rangle$

definition *enabled-projection-Inf* :: $(f, v) \text{ AF inference set} \Rightarrow v \text{ total-interpretation} \Rightarrow$
 $f \text{ inference set}$ $(\text{infix } \iota \text{proj}_J \ 60)$ **where**
 $\langle I \ \iota \text{proj}_J \ J = \{\iota F\text{-of } \iota \mid \iota. \iota \in I \wedge \text{enabled-inf } \iota \ J\} \rangle$

fun *fml-ext* :: $v \text{ sign} \Rightarrow f \text{ sign}$ **where**
 $\text{fml-ext } (\text{Pos } v) = \text{Pos } (\text{fml } v) \mid$
 $\text{fml-ext } (\text{Neg } v) = \text{Neg } (\text{fml } v)$

lemma *fml-ext-is-mapping*: $\langle \text{fml-ext } v = \text{map-sign } \text{fml } v \rangle$
 $\langle \text{proof} \rangle$

lemma *fml-ext-preserves-sign*: $\text{is-Pos } v \equiv \text{is-Pos } (\text{fml-ext } v)$
 $\langle \text{proof} \rangle$

lemma *to-V-fml-ext [simp]*: $\langle \text{to-V } (fml\text{-ext } v) = fml \text{ (to-V } v) \rangle$
 $\langle \text{proof} \rangle$

lemma *fml-ext-preserves-val*: $\langle \text{to-V } v1 = \text{to-V } v2 \implies \text{to-V } (fml\text{-ext } v1) = \text{to-V } (fml\text{-ext } v2) \rangle$
 $\langle \text{proof} \rangle$

definition *sound-consistent* :: $'v \text{ total-interpretation} \Rightarrow \text{bool}$ **where**
 $\langle \text{sound-consistent } J \equiv \neg (\text{sound-cons. entails-neg } (fml\text{-ext } ' (total\text{-strip } J)) \{Pos \text{ bot}\}) \rangle$

definition *propositional-model* :: $'v \text{ total-interpretation} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow \text{bool}$
 $(\text{infix } \models_p 50)$ **where**
 $\langle J \models_p \mathcal{N} \equiv \text{bot} \notin ((proj_{\perp} \mathcal{N}) \text{ proj}_J J) \rangle$

lemma $\langle J \models_p \{\} \rangle$
 $\langle \text{proof} \rangle$

The definition below is essentially the same as the one above since $\text{term}(proj_{\perp} \mathcal{N}) \text{ proj}_J J$ is either empty or contains only bot

definition *propositional-model2* :: $'v \text{ total-interpretation} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow \text{bool}$
 $(\text{infix } \models_{p2} 50)$ **where**
 $\langle J \models_{p2} \mathcal{N} \equiv (\{\} = ((proj_{\perp} \mathcal{N}) \text{ proj}_J J)) \rangle$

lemma *subset-model-p2*: $\langle \mathcal{N}' \subseteq \mathcal{N} \implies J \models_{p2} \mathcal{N} \implies J \models_{p2} \mathcal{N}' \rangle$
 $\langle \text{proof} \rangle$

lemma *subset-not-model*: $\langle \neg J \models_{p2} \mathcal{N} \implies \mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2 \implies J \models_{p2} \mathcal{N}_1 \implies \neg J \models_{p2} \mathcal{N}_2 \rangle$
 $\langle \text{proof} \rangle$

lemma *supset-not-model-p2*: $\langle \mathcal{N}' \subseteq \mathcal{N} \implies \neg J \models_{p2} \mathcal{N}' \implies \neg J \models_{p2} \mathcal{N} \rangle$
 $\langle \text{proof} \rangle$

fun *sign-to-atomic-formula* :: $'v \text{ sign} \Rightarrow 'v \text{ formula}$ **where**
 $\langle \text{sign-to-atomic-formula } (Pos \ v) = Atom \ v \mid$
 $\langle \text{sign-to-atomic-formula } (Neg \ v) = Not \ (Atom \ v) \rangle$

definition *sign-set-to-formula-set* :: $'v \text{ sign set} \Rightarrow 'v \text{ formula set}$ **where**
 $\langle \text{sign-set-to-formula-set } A = \text{sign-to-atomic-formula } ' A \rangle$

lemma *form-shape-sign-set*: $\langle \forall f \in \text{sign-set-to-formula-set } A. \exists v. f = Atom \ v \vee f = Not \ (Atom \ v) \rangle$
 $\langle \text{proof} \rangle$

definition *AF-to-formula-set* :: $('f, 'v) \text{ AF} \Rightarrow 'v \text{ formula set}$ **where**

$\langle \text{AF-to-formula-set } \mathcal{C} = \text{sign-set-to-formula-set } (\text{neg } ' \text{ fset } (A\text{-of } \mathcal{C})) \rangle$

definition $AF\text{-to-formula} :: ('f, 'v) AF \Rightarrow 'v \text{ formula}$ **where**

$\langle AF\text{-to-formula } C = \text{BigOr } (\text{map sign-to-atomic-formula } (\text{map neg } (\text{list-of-fset } (A\text{-of } C)))) \rangle$

lemma $\text{form-shape-}AF: \langle \forall f \in AF\text{-to-formula-set } C. \exists v. f = \text{Atom } v \vee f = \text{Not } (\text{Atom } v) \rangle$
 $\langle \text{proof} \rangle$

definition $AF\text{-proj-to-formula-set-set} :: ('f, 'v) AF \text{ set} \Rightarrow 'v \text{ formula set set}$ **where**

$\langle AF\text{-proj-to-formula-set-set } \mathcal{N} = AF\text{-to-formula-set } '(\text{proj}_{\perp} \mathcal{N}) \rangle$

definition $AF\text{-proj-to-formula-set} :: ('f, 'v) AF \text{ set} \Rightarrow 'v \text{ formula set}$ **where**
 $\langle AF\text{-proj-to-formula-set } \mathcal{N} = AF\text{-to-formula } '(\text{proj}_{\perp} \mathcal{N}) \rangle$

definition $AF\text{-assertions-to-formula} :: ('f, 'v) AF \Rightarrow 'v \text{ formula}$ **where**

$\langle AF\text{-assertions-to-formula } C = \text{BigAnd } (\text{map sign-to-atomic-formula } (\text{list-of-fset } (A\text{-of } C))) \rangle$

definition $AF\text{-assertions-to-formula-set} :: ('f, 'v) AF \text{ set} \Rightarrow 'v \text{ formula set}$ **where**
 $\langle AF\text{-assertions-to-formula-set } \mathcal{N} = AF\text{-assertions-to-formula } ' \mathcal{N} \rangle$

lemma $F\text{-to-}C\text{-set}: \langle \forall F \in AF\text{-proj-to-formula-set-set } \mathcal{N}. \exists C \in \text{proj}_{\perp} \mathcal{N}. F = \text{sign-to-atomic-formula } ' \text{neg } ' \text{fset } (A\text{-of } C) \rangle$
 $\langle \text{proof} \rangle$

lemma $F\text{-to-}C: \langle \forall F \in AF\text{-proj-to-formula-set } \mathcal{N}. \exists C \in \text{proj}_{\perp} \mathcal{N}. F = \text{BigOr } (\text{map sign-to-atomic-formula } (\text{map neg } (\text{list-of-fset } (A\text{-of } C)))) \rangle$
 $\langle \text{proof} \rangle$

lemma $C\text{-to-}F\text{-set}: \langle \forall C \in \text{proj}_{\perp} \mathcal{N}. \exists F \in AF\text{-proj-to-formula-set-set } \mathcal{N}. F = \text{sign-to-atomic-formula } ' \text{neg } ' \text{fset } (A\text{-of } C) \rangle$
 $\langle \text{proof} \rangle$

lemma $C\text{-to-}F: \langle \forall C \in \text{proj}_{\perp} \mathcal{N}. \exists F \in AF\text{-proj-to-formula-set } \mathcal{N}. F = \text{BigOr } (\text{map sign-to-atomic-formula } (\text{map neg } (\text{list-of-fset } (A\text{-of } C)))) \rangle$
 $\langle \text{proof} \rangle$

lemma $\text{form-shape-proj}: \langle \forall f \in \bigcup (AF\text{-proj-to-formula-set-set } \mathcal{N}). \exists v. f = \text{Atom } v \vee f = \text{Not } (\text{Atom } v) \rangle$
 $\langle \text{proof} \rangle$

definition $\text{to-valuation} :: 'v \text{ total-interpretation} \Rightarrow 'v \text{ valuation}$ **where**
 $\langle \text{to-valuation } J = (\lambda a. \text{Pos } a \in_t J) \rangle$

lemma $\text{val-strip-pos}: \langle \text{to-valuation } J \ a \equiv \text{Pos } a \in \text{total-strip } J \rangle$
 $\langle \text{proof} \rangle$

lemma $\text{val-strip-neg}: \langle (\neg \text{to-valuation } J \ a) = (\text{Neg } a \in \text{total-strip } J) \rangle$

$\langle \text{proof} \rangle$

lemma *equiv-prop-entails*: $\langle (J \models_p \mathcal{N}) \longleftrightarrow (J \models_p^2 \mathcal{N}) \rangle$
 $\langle \text{proof} \rangle$

definition *propositional-model3* :: $'v$ total-interpretation $\Rightarrow ('f, 'v)$ AF set $\Rightarrow$ bool
 (infix $\models_p^3$ 50) **where**
 $\langle J \models_p^3 \mathcal{N} \equiv (\forall F \in \text{AF-proj-to-formula-set-set } \mathcal{N}. \exists f \in F. \text{formula-antics (to-valuation } J) f) \rangle$

lemma *equiv-prop-entail2-sema*:
 $\langle (J \models_p^2 \mathcal{N}) \longleftrightarrow (J \models_p^3 \mathcal{N}) \rangle$
 $\langle \text{proof} \rangle$

lemma *equiv-prop-entail2-sema2*:
 $\langle (J \models_p^2 \mathcal{N}) \longleftrightarrow (\forall F \in \text{AF-proj-to-formula-set } \mathcal{N}. \text{formula-antics (to-valuation } J) F) \rangle$
 $\langle \text{proof} \rangle$

lemma *equiv-prop-entail-sema*:
 $\langle (J \models_p \mathcal{N}) \longleftrightarrow (J \models_p^3 \mathcal{N}) \rangle$
 $\langle \text{proof} \rangle$

lemma $\langle f \text{ ' fset } A = \text{set (map } f \text{ (list-of-fset } A)) \rangle$
 $\langle \text{proof} \rangle$

lemma *equiv-prop-sema1-sema2*:
 $\langle (J \models_p^3 \mathcal{N}) \longleftrightarrow$
 $(\forall F \in \text{AF-proj-to-formula-set } \mathcal{N}. \text{formula-antics (to-valuation } J) F) \rangle$
 $\langle \text{proof} \rangle$

lemma *equiv-enabled-assertions-sema*:
 $\langle (\text{enabled-set } \mathcal{N} J) \longleftrightarrow (\forall F \in \text{AF-assertions-to-formula-set } \mathcal{N}. \text{formula-antics (to-valuation } J) F) \rangle$
 $\langle \text{proof} \rangle$

definition *sound-propositional-model* :: $'v$ total-interpretation $\Rightarrow ('f, 'v)$ AF set
 $\Rightarrow$ bool
 (infix $\models_{s_p}$ 50) **where**
 $\langle J \models_{s_p} \mathcal{N} \equiv (\text{bot} \notin ((\text{enabled-projection (propositional-projection } \mathcal{N}) J)) \vee$
 $\neg \text{sound-consistent } J) \rangle$

definition *propositionally-unsatisfiable* :: $('f, 'v)$ AF set $\Rightarrow$ bool **where**
 $\langle \text{propositionally-unsatisfiable } \mathcal{N} \equiv \forall J. \neg (J \models_p \mathcal{N}) \rangle$

lemma *unsat-simp*:

assumes

$\langle \neg \text{sat } (S' \cup S :: 'v \text{ formula set}) \rangle$
 $\langle \text{sat } S' \rangle$
 $\langle \bigcup (\text{atoms } ' S') \cap \bigcup (\text{atoms } ' S) = \{\} \rangle$

shows

$\langle \neg \text{sat } S \rangle$
 $\langle \text{proof} \rangle$

lemma *proj-to-form-un*: $\langle \text{AF-proj-to-formula-set } (A \cup B) =$

$\text{AF-proj-to-formula-set } A \cup \text{AF-proj-to-formula-set } B \rangle$

$\langle \text{proof} \rangle$

lemma *unsat-AF-simp*:

assumes

$\langle \neg \text{sat } (\text{AF-proj-to-formula-set } (S' \cup S)) \rangle$
 $\langle \text{sat } (\text{AF-proj-to-formula-set } S') \rangle$
 $\langle \bigcup (\text{atoms } ' (\text{AF-proj-to-formula-set } S')) \cap \bigcup (\text{atoms } ' (\text{AF-proj-to-formula-set } S)) = \{\} \rangle$

shows

$\langle \neg \text{sat } (\text{AF-proj-to-formula-set } S) \rangle$
 $\langle \text{proof} \rangle$

lemma *set-list-of-fset[simp]*: $\langle \text{set } (\text{list-of-fset } A) = \text{fset } A \rangle$

$\langle \text{proof} \rangle$

lemma *vars-in-assertion*: $\langle \text{to-}V ' (\text{set } (\text{list-of-fset } A)) = \text{to-}V ' (\text{fset } A) \rangle$

$\langle \text{proof} \rangle$

lemma *atoms-bigor*: $\langle \text{atoms } (\text{BigOr } L) = \bigcup (\text{atoms } ' (\text{set } L)) \rangle$

$\langle \text{proof} \rangle$

lemma *atoms-neg*: $\langle \text{atoms } (\text{sign-to-atomic-formula } (\text{neg } A)) = \text{atoms } (\text{sign-to-atomic-formula } A) \rangle$

$\langle \text{proof} \rangle$

lemma *set-maps-list-of-fset*: $\langle \text{set } (\text{map sign-to-atomic-formula } (\text{map neg } (\text{list-of-fset } A))) =$

$\text{sign-to-atomic-formula } ' \text{neg } ' \text{fset } A \rangle$

$\langle \text{proof} \rangle$

lemma *atoms-to-V-mono*: $\langle \text{atoms } (\text{sign-to-atomic-formula } A) = \{\text{to-}V A\} \rangle$

$\langle \text{proof} \rangle$

lemma *atoms-to-V*: $\langle \bigcup (\text{atoms } ' \text{sign-to-atomic-formula } ' A) = \text{to-}V ' A \rangle$

$\langle \text{proof} \rangle$

lemma *atoms-to-V-AF*: $\langle \text{atoms } (\text{AF-to-formula } (\text{Pair } C A)) = \text{to-}V ' (\text{fset } A) \rangle$

$\langle \text{proof} \rangle$

lemma *atoms-to-V-A-of*: $\langle \text{atoms } (AF\text{-to-formula } \mathcal{C}) = \text{to-V } (fset (A\text{-of } \mathcal{C})) \rangle$
 $\langle \text{proof} \rangle$

lemma *atoms-to-V-un*: $\langle \bigcup (\text{atoms } (AF\text{-to-formula } \mathcal{S})) = \bigcup \{ \text{to-V } (fset A) \mid A. A \in A\text{-of } \mathcal{S} \} \rangle$
 $\langle \text{proof} \rangle$

lemma *atoms-simp*: $\langle \bigcup (\text{atoms } (AF\text{-proj-to-formula-set } S)) = \text{to-V } (fset (A\text{-of } (proj_{\perp} S))) \rangle$
 $\langle \text{proof} \rangle$

lemma *val-from-interp*: $\langle \forall \mathcal{A}. \exists J. \mathcal{A} = \text{to-valuation } J \rangle$
 $\langle \text{proof} \rangle$

lemma *interp-from-val*: $\langle \forall J. \exists \mathcal{A}. \mathcal{A} = \text{to-valuation } J \rangle$
 $\langle \text{proof} \rangle$

lemma *compactness-unsat*: $\langle (\neg \text{sat } (S::'v \text{ formula set})) \longleftrightarrow (\exists s \subseteq S. \text{finite } s \wedge \neg \text{sat } s) \rangle$
 $\langle \text{proof} \rangle$

lemma *never-enabled-finite-subset*:
 $\langle \forall J. \neg \text{enabled-set } \mathcal{N} \ J \implies \exists \mathcal{N}' \subseteq \mathcal{N}. \text{finite } \mathcal{N}' \wedge (\forall J. \neg \text{enabled-set } \mathcal{N}' \ J) \rangle$
 $\langle \text{proof} \rangle$

lemma *compactness-AF-proj*: $\langle (\forall J. \neg J \models_p^2 \mathcal{N}) \longleftrightarrow (\exists \mathcal{N}' \subseteq \mathcal{N}. \text{finite } \mathcal{N}' \wedge (\forall J. \neg J \models_p^2 \mathcal{N}')) \rangle$
 $\langle \text{proof} \rangle$

lemma *prop-unsat-compactness*:
 $\langle \text{propositionally-unsatisfiable } A \implies \exists B \subseteq A. \text{finite } B \wedge \text{propositionally-unsatisfiable } B \rangle$
 $\langle \text{proof} \rangle$

definition $\mathcal{E}\text{-from} :: \langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \rangle$ **where**
 $\langle \mathcal{E}\text{-from } \mathcal{N} \equiv \{ \text{Pair bot } \{ | \text{neg } a | \} \mid a. \exists \mathcal{C} \in \mathcal{N}. a \in fset (A\text{-of } \mathcal{C}) \} \rangle$

lemma *prop-proj-E-from*: $\langle proj_{\perp} (\mathcal{E}\text{-from } \mathcal{N}) = \mathcal{E}\text{-from } \mathcal{N} \rangle$
 $\langle \text{proof} \rangle$

lemma *prop-proj-sub*: $\langle proj_{\perp} \mathcal{N} = \mathcal{N} \implies \mathcal{N}' \subseteq \mathcal{N} \implies proj_{\perp} \mathcal{N}' = \mathcal{N}' \rangle$
 $\langle \text{proof} \rangle$

lemma *prop-proj-distrib*: $\langle proj_{\perp} (A \cup B) = proj_{\perp} A \cup proj_{\perp} B \rangle$
 $\langle \text{proof} \rangle$

lemma *v-in-E*: $\langle \text{Pair bot } \{| \text{Pos } v | \} \in \mathcal{E}\text{-from } \mathcal{N} \vee \text{Pair bot } \{| \text{Neg } v | \} \in \mathcal{E}\text{-from } \mathcal{N} \implies$
 $\exists \mathcal{C} \in \mathcal{N}. v \in \text{to-}V \text{ ' (fset (A-of } \mathcal{C})) \rangle$
 $\langle \text{proof} \rangle$

lemma *a-in-E*: $\langle \exists J. J \models_p \mathcal{E}\text{-from } \mathcal{N} \implies \text{Pair bot } \{| \text{neg } a | \} \in \mathcal{E}\text{-from } \mathcal{N} \implies$
 $\neg (\text{Pair bot } \{| a | \} \in \mathcal{E}\text{-from } \mathcal{N}) \rangle$
 $\langle \text{proof} \rangle$

lemma *equiv-E-enabled-N*:
shows $\langle J \models_p \mathcal{E}\text{-from } \mathcal{N} \longleftrightarrow \text{enabled-set } \mathcal{N} \ J \rangle$
 $\langle \text{proof} \rangle$

definition *AF-entails* :: $(f, v) \text{ AF set} \Rightarrow (f, v) \text{ AF set} \Rightarrow \text{bool}$ (**infix** $\models_{AF}$ 50)
where
 $\langle \text{AF-entails } \mathcal{M} \ \mathcal{N} \equiv (\forall J. (\text{enabled-set } \mathcal{N} \ J \longrightarrow \mathcal{M} \ \text{proj}_J \ J \models F\text{-of ' } \mathcal{N})) \rangle$

lemma *prop-unsat-to-AF-entails-bot*: $\langle \text{propositionally-unsatisfiable } \mathcal{M} \implies \mathcal{M} \models_{AF} \{ \text{to-AF bot} \} \rangle$
 $\langle \text{proof} \rangle$

lemma $\langle \text{enabled-set } \{ \} \ J \rangle$
 $\langle \text{proof} \rangle$

lemma $\langle (\forall J. \neg (\text{enabled-set } \mathcal{N} \ J)) \implies (\mathcal{M} \models_{AF} \mathcal{N}) \rangle$
 $\langle \text{proof} \rangle$

definition *AF-entails-sound* :: $(f, v) \text{ AF set} \Rightarrow (f, v) \text{ AF set} \Rightarrow \text{bool}$ (**infix** $\models_{sAF}$ 50) **where**
 $\langle \text{AF-entails-sound } \mathcal{M} \ \mathcal{N} \equiv (\forall J. (\text{enabled-set } \mathcal{N} \ J \longrightarrow$
 $\text{sound-cons.entails-neg } ((\text{fml-ext ' (total-strip } J)) \cup (\text{Pos ' } (\mathcal{M} \ \text{proj}_J \ J))) (\text{Pos ' } F\text{-of ' } \mathcal{N}))) \rangle$

lemma *distrib-proj*: $\langle \mathcal{M} \cup \mathcal{N} \ \text{proj}_J \ J = (\mathcal{M} \ \text{proj}_J \ J) \cup (\mathcal{N} \ \text{proj}_J \ J) \rangle$
 $\langle \text{proof} \rangle$

lemma *distrib-proj-singleton*: $\langle \mathcal{M} \cup \{ \mathcal{C} \} \ \text{proj}_J \ J = (\mathcal{M} \ \text{proj}_J \ J) \cup (\{ \mathcal{C} \} \ \text{proj}_J \ J) \rangle$
 $\langle \text{proof} \rangle$

lemma *enabled-union2*: $\langle \text{enabled-set } (\mathcal{M} \cup \mathcal{N}) \ J \implies \text{enabled-set } \mathcal{N} \ J \rangle$
 $\langle \text{proof} \rangle$

lemma *enabled-union1*: $\langle \text{enabled-set } (\mathcal{M} \cup \mathcal{N}) \ J \implies \text{enabled-set } \mathcal{M} \ J \rangle$
 $\langle \text{proof} \rangle$

lemma *finite-subset-image-strong*:
assumes *finite U* **and**
 $(\forall C \in U. (\exists D \in W. P \ D = C \wedge Q \ D))$
shows $\exists W' \subseteq W. \text{finite } W' \wedge U = P \text{ ' } W' \wedge (\forall D' \in W'. Q \ D')$

$\langle \text{proof} \rangle$

lemma *all-to-ex*: $\langle \forall x. P\ x \implies \exists x. P\ x \rangle$ **for** P $\langle \text{proof} \rangle$

lemma *three-skolems*:

assumes $\langle \bigwedge U. P\ U \implies \exists X\ Y\ Z. Q\ U\ X\ Y\ Z \rangle$

shows $\langle (\bigwedge X\text{-of}\ Y\text{-of}\ Z\text{-of}. (\bigwedge U. P\ U \implies Q\ U\ (X\text{-of}\ U)\ (Y\text{-of}\ U)\ (Z\text{-of}\ U))) \implies \text{thesis} \rangle \implies \text{thesis}$
 $\langle \text{proof} \rangle$

lemma *finite-subset-with-prop*:

assumes $\langle \exists Js. A = f\ ' Js \wedge (\forall J \in Js. P\ J) \rangle$ **and**

$\langle \text{finite}\ C \rangle$ **and**

$\langle B = C \cap A \rangle$

shows $\langle \exists Js. B = f\ ' Js \wedge (\forall J \in Js. P\ J) \wedge \text{finite}\ Js \rangle$

$\langle \text{proof} \rangle$

lemma *to-V-neg [simp]*: $\langle \text{to-V}\ (\text{neg}\ a) = \text{to-V}\ a \rangle$

$\langle \text{proof} \rangle$

sublocale *AF-cons-rel: consequence-relation to-AF bot AF-entails*

$\langle \text{proof} \rangle$

sublocale *neg-ext-sound-cons-rel: consequence-relation Pos bot sound-cons.entails-neg*

$\langle \text{proof} \rangle$

lemma *AF-ext-sound-cons-rel*: $\langle \text{consequence-relation}\ (\text{to-AF}\ \text{bot})\ \text{AF-entails-sound} \rangle$

$\langle \text{proof} \rangle$

interpretation *AF-sound-cons-rel: consequence-relation to-AF bot AF-entails-sound*

$\langle \text{proof} \rangle$

lemma *f-of-to-AF [simp]*: $\langle F\text{-of}\ ' \text{to-AF}\ ' N = N \rangle$

$\langle \text{proof} \rangle$

lemma *to-AF-proj-J [simp]*: $\langle \text{to-AF}\ ' M\ \text{proj}_J\ J = M \rangle$

$\langle \text{proof} \rangle$

lemma *enabled-to-AF-set [simp]*: $\langle \text{enabled-set}\ (\text{to-AF}\ ' N)\ J \rangle$

$\langle \text{proof} \rangle$

lemma *pos-not-pos-empty [simp]*: $\langle \{ \text{to-V}\ C \mid C. C \in \text{Pos}\ ' N \wedge \neg \text{is-Pos}\ C \} = \{ \} \rangle$

$\langle \text{proof} \rangle$

lemma *pos-not-pos-simp [simp]*:

$\langle \{ \text{to-V}\ C \mid C. C \in U \cup \text{Pos}\ ' M \wedge \neg \text{is-Pos}\ C \} = \{ \text{to-V}\ C \mid C. C \in U \wedge \neg \text{is-Pos}\ C \} \rangle$

$\langle \text{proof} \rangle$

lemma *pos-pos-simp* [simp]: $\langle \{ \text{to-}V\ C \mid C. C \in \text{Pos} \text{ ' } F\text{-of ' } N \wedge \text{is-Pos } C \} = F\text{-of ' } N \rangle$
 $\langle \text{proof} \rangle$

lemma *proj-F-of* [simp]: $\langle \{ C. F\text{-of } C \in M \} \text{ proj}_J\ J = M \rangle$
 $\langle \text{proof} \rangle$

lemma *f-of-F-of* [simp]: $\langle F\text{-of ' } \{ C. F\text{-of } C \in M \} = M \rangle$
 $\langle \text{proof} \rangle$

lemma *set-on-union-triple-split*: $\langle \{ f\ C \mid C. C \in M \cup N \cup g\ J \wedge l\ C\ J \} = \{ f\ C \mid C. C \in M \wedge l\ C\ J \} \cup \{ f\ C \mid C. C \in N \wedge l\ C\ J \} \cup \{ f\ C \mid C. C \in g\ J \wedge l\ C\ J \} \rangle$
 $\langle \text{proof} \rangle$

lemma *not-enabled-enabled-empty* [simp]:
 $\langle \{ F\text{-of } C \mid C. C \in \{ C. F\text{-of } C \in Q' \wedge \neg \text{enabled } C\ J \} \wedge \text{enabled } C\ J \} = \{ \} \rangle$
 $\langle \text{proof} \rangle$

lemma *f-of-simp-enabled* [simp]: $\langle \{ F\text{-of } C \mid C. F\text{-of } C \in M \wedge \text{enabled } C\ J \} = M \rangle$
 $\langle \text{proof} \rangle$

lemma *f-of-enabled-simp* [simp]: $\langle F\text{-of ' } \{ C. F\text{-of } C \in M \wedge \text{enabled } C\ J \} = M \rangle$
 $\langle \text{proof} \rangle$

lemma $\langle (\text{to-}AF \text{ ' } M \models_{AF} \text{to-}AF \text{ ' } N) \equiv (M \models N) \rangle$
 $\langle \text{proof} \rangle$

sublocale *ext-cons-rel-std*: *consequence-relation Pos (to-}AF bot) AF-cons-rel.entails-neg*
 $\langle \text{proof} \rangle$

sublocale *sound-cons-rel*: *consequence-relation Pos bot sound-cons.entails-neg*
 $\langle \text{proof} \rangle$

lemma *pos-in-pos-simp* [simp]: $\langle \{ C. \text{Pos } C \in \text{Pos ' } N \} = N \rangle \langle \text{proof} \rangle$

lemma *neg-not-in-pos-simp* [simp]: $\langle \{ C. \text{Neg } C \in \text{Pos ' } N \} = \{ \} \rangle \langle \text{proof} \rangle$

lemma *neg-in-pos-simp* [simp]: $\langle \{ C. \text{Neg } C \in P \vee \text{Neg } C \in \text{Pos ' } M \} = \{ C. \text{Neg } C \in P \} \rangle \langle \text{proof} \rangle$

lemma *pos-in-pos-partial-simp* [simp]: $\langle \{ C. \text{Pos } C \in P \vee \text{Pos } C \in \text{Pos ' } M \} = \{ C. \text{Pos } C \in P \} \cup M \rangle \langle \text{proof} \rangle$

lemma $\langle (\text{to-}AF \text{ ' } M \models_{sAF} \text{to-}AF \text{ ' } N) \equiv (M \models_s N) \rangle$
 $\langle \text{proof} \rangle$

lemma *strong-entails-bot-cases*: $\langle \mathcal{N} \cup \{ AF.\text{Pair bot } A \} \models_{sAF} \{ AF.\text{Pair bot } B \} \rangle$

$\implies$
 $\langle \forall J. \text{fset } B \subseteq \text{total-strip } J \longrightarrow$
 $(\text{fml-ext } \text{'total-strip } J \cup \text{Pos } \text{' } (\mathcal{N} \text{ proj}_J J)) \models_{s\sim} \{\text{Pos bot}\} \vee \text{fset } A \subseteq \text{total-strip}$
 $J) \rangle$
 $\langle \text{proof} \rangle$

lemma *strong-entails-bot-cases-Union:*

$\langle \mathcal{N} \cup \mathcal{M} \models_{sAF} \{AF.Pair \text{ bot } B\} \implies (\forall x \in \mathcal{M}. F\text{-of } x = \text{bot}) \implies$
 $(\forall J. \text{fset } B \subseteq \text{total-strip } J \longrightarrow$
 $(\text{fml-ext } \text{'total-strip } J \cup \text{Pos } \text{' } (\mathcal{N} \text{ proj}_J J)) \models_{s\sim} \{\text{Pos bot}\} \vee$
 $(\exists A \in A\text{-of } \text{' } \mathcal{M}. \text{fset } A \subseteq \text{total-strip } J)) \rangle$
 $\langle \text{proof} \rangle$

lemma *AF-entails-sound-right-disjunctive:* $\langle (\exists C' \in A. \mathcal{M} \models_{sAF} \{C'\}) \implies \mathcal{M} \models_{sAF} A \rangle$
 $\langle \text{proof} \rangle$

6.1 Local saturation

To fully capture completeness for splitting, we need to use weaker notions of saturation and fairness.

definition *locally-saturated* :: $\langle \text{'f, 'v} \rangle \text{ AF set} \Rightarrow \text{bool} \rangle$ **where**

$\langle \text{locally-saturated } \mathcal{N} \equiv$
 $\text{to-}AF \text{ bot} \in \mathcal{N} \vee$
 $(\exists J :: \text{'v total-interpretation. } J \models_p \mathcal{N} \wedge \text{saturated } (\mathcal{N} \text{ proj}_J J)) \rangle$

definition *locally-fair* :: $\langle \text{'f, 'v} \rangle \text{ AF set infinite-llist} \Rightarrow \text{bool} \rangle$ **where**

$\langle \text{locally-fair } \mathcal{N} i \equiv$
 $(\exists i. \text{to-}AF \text{ bot} \in \text{llnth } \mathcal{N} i i)$
 $\vee (\exists J :: \text{'v total-interpretation. } J \models_p \text{lim-inf } \mathcal{N} i \wedge$
 $\text{Inf-from } (\text{lim-inf } \mathcal{N} i \text{ proj}_J J) \subseteq (\bigcup i. \text{Red}_I (\text{llnth } \mathcal{N} i i \text{ proj}_J J))) \rangle$

end

locale *strong-statically-complete-annotated-calculus* =

$\text{calculus-with-annotated-consrel } \text{bot } \text{Inf entails entails-sound } \text{Red}_I \text{ Red}_F \text{ fml asn}$
 $+ S\text{-calculus: calculus to-}AF \text{ bot } S\text{Inf } AF\text{-entails } S\text{Red}_I \text{ SRed}_F$

for

$\text{bot} :: \text{'f and}$
 $\text{Inf} :: \langle \text{'f inference set} \rangle \text{ and}$
 $\text{entails} :: \text{'f set} \Rightarrow \text{'f set} \Rightarrow \text{bool (infix } \models 50) \text{ and}$
 $\text{entails-sound} :: \text{'f set} \Rightarrow \text{'f set} \Rightarrow \text{bool (infix } \models_s 50) \text{ and}$
 $\text{Red}_I :: \text{'f set} \Rightarrow \text{'f inference set and}$
 $\text{Red}_F :: \text{'f set} \Rightarrow \text{'f set and}$
 $\text{fml} :: \langle \text{'v} :: \text{countable} \Rightarrow \text{'f} \rangle \text{ and}$
 $\text{asn} :: \langle \text{'f sign} \Rightarrow \text{'v sign set} \rangle \text{ and}$

```

SInf :: ('f, 'v) AF inference set and
SRedI :: ('f, 'v) AF set ⇒ ('f, 'v) AF inference set and
SRedF :: ('f, 'v) AF set ⇒ ('f, 'v) AF set
+ assumes
  strong-static-completeness: ⟨locally-saturated N ⇒ N ⊨AF {to-AF bot} ⇒
to-AF bot ∈ N⟩

locale strong-dynamically-complete-annotated-calculus =
  calculus-with-annotated-consrel bot Inf entails entails-sound RedI RedF fml asn
+ S-calculus: calculus to-AF bot SInf AF-entails SRedI SRedF
for
  bot :: 'f and
  Inf :: ⟨'f inference set⟩ and
  entails :: 'f set ⇒ 'f set ⇒ bool (infix ⊨ 50) and
  entails-sound :: 'f set ⇒ 'f set ⇒ bool (infix ⊨s 50) and
  RedI :: 'f set ⇒ 'f inference set and
  RedF :: 'f set ⇒ 'f set and
  fml :: ⟨'v :: countable ⇒ 'f⟩ and
  asn :: ⟨'f sign ⇒ 'v sign set⟩ and
  SInf :: ('f, 'v) AF inference set and
  SRedI :: ('f, 'v) AF set ⇒ ('f, 'v) AF inference set and
  SRedF :: ('f, 'v) AF set ⇒ ('f, 'v) AF set
+ assumes
  strong-dynamic-completeness: ⟨is-derivation S-calculus.derive N i ⇒ locally-fair
N i ⇒
    llhd N i ⊨AF {to-AF bot} ⇒ ∃ i. to-AF bot ∈ llth N i⟩

```

end

7 Lifting to Non-ground Calculi

The section 3.1 to 3.3 of the report are covered by the current section. Various forms of lifting are proven correct. These allow to obtain the dynamic refutational completeness of a non-ground calculus from the static refutational completeness of its ground counterpart.

```

theory Light-Lifting-to-Non-Ground-Calculi
imports
  Saturation-Framework.Intersection-Calculus
  Saturation-Framework.Calculus-Variations
  Well-Quasi-Orders.Minimal-Elements
begin

```

7.1 Standard Lifting

```

locale light-standard-lifting = inference-system Inf-F +
  ground: calculus Bot-G Inf-G entails-G Red-I-G Red-F-G
for

```

$Inf-F :: \langle 'f \text{ inference set} \rangle$ **and**
 $Bot-G :: \langle 'g \text{ set} \rangle$ **and**
 $Inf-G :: \langle 'g \text{ inference set} \rangle$ **and**
 $entails-G :: \langle 'g \text{ set} \Rightarrow 'g \text{ set} \Rightarrow \text{bool} \rangle$ (**infix** $\models_G 50$) **and**
 $Red-I-G :: \langle 'g \text{ set} \Rightarrow 'g \text{ inference set} \rangle$ **and**
 $Red-F-G :: \langle 'g \text{ set} \Rightarrow 'g \text{ set} \rangle$
+ fixes
 $Bot-F :: \langle 'f \text{ set} \rangle$ **and**
 $\mathcal{G}-F :: \langle 'f \Rightarrow 'g \text{ set} \rangle$ **and**
 $\mathcal{G}-I :: \langle 'f \text{ inference} \Rightarrow 'g \text{ inference set option} \rangle$
assumes
 $Bot-F\text{-not-empty}: Bot-F \neq \{\}$ **and**
 $Bot\text{-map-not-empty}: \langle B \in Bot-F \Longrightarrow \mathcal{G}-F B \neq \{\} \rangle$ **and**
 $Bot\text{-map}: \langle B \in Bot-F \Longrightarrow \mathcal{G}-F B \subseteq Bot-G \rangle$ **and**
 $inf\text{-map}: \langle \iota \in Inf-F \Longrightarrow \mathcal{G}-I \iota \neq None \Longrightarrow the (\mathcal{G}-I \iota) \subseteq Red-I-G (\mathcal{G}-F (concl\text{-of} \iota)) \rangle$
begin

abbreviation $\mathcal{G}\text{-Fset} :: \langle 'f \text{ set} \Rightarrow 'g \text{ set} \rangle$ **where**
 $\langle \mathcal{G}\text{-Fset } N \equiv \bigcup (\mathcal{G}\text{-F } N) \rangle$

lemma $\mathcal{G}\text{-subset}: \langle N1 \subseteq N2 \Longrightarrow \mathcal{G}\text{-Fset } N1 \subseteq \mathcal{G}\text{-Fset } N2 \rangle$ $\langle proof \rangle$

abbreviation $entails\text{-}\mathcal{G} :: \langle 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow \text{bool} \rangle$ (**infix** $\models_{\mathcal{G}} 50$) **where**
 $\langle N1 \models_{\mathcal{G}} N2 \equiv \mathcal{G}\text{-Fset } N1 \models_G \mathcal{G}\text{-Fset } N2 \rangle$

lemma $subs\text{-}Bot\text{-}G\text{-}entails$:
assumes
 $not\text{-empty}: \langle sB \neq \{\} \rangle$ **and**
 $in\text{-bot}: \langle sB \subseteq Bot-G \rangle$
shows $\langle sB \models_G N \rangle$
 $\langle proof \rangle$

sublocale $consequence\text{-}relation \ Bot-F \ entails\text{-}\mathcal{G}$
 $\langle proof \rangle$

definition $Red\text{-}I\text{-}\mathcal{G} :: 'f \text{ set} \Rightarrow 'f \text{ inference set}$ **where**
 $\langle Red\text{-}I\text{-}\mathcal{G} N = \{ \iota \in Inf-F. (\mathcal{G}\text{-I } \iota \neq None \wedge the (\mathcal{G}\text{-I } \iota) \subseteq Red\text{-}I\text{-}G (\mathcal{G}\text{-Fset } N)) \vee (\mathcal{G}\text{-I } \iota = None \wedge \mathcal{G}\text{-F } (concl\text{-of } \iota) \subseteq \mathcal{G}\text{-Fset } N \cup Red\text{-}F\text{-}G (\mathcal{G}\text{-Fset } N)) \} \rangle$

definition $Red\text{-}F\text{-}\mathcal{G} :: 'f \text{ set} \Rightarrow 'f \text{ set}$ **where**
 $\langle Red\text{-}F\text{-}\mathcal{G} N = \{ C. \forall D \in \mathcal{G}\text{-F } C. D \in Red\text{-}F\text{-}G (\mathcal{G}\text{-Fset } N) \} \rangle$
end

7.2 Strong Standard Lifting

locale $light\text{-}strong\text{-}standard\text{-}lifting = inference\text{-}system \ Inf-F +$

```

ground: calculus Bot-G Inf-G entails-G Red-I-G Red-F-G
for
  Inf-F :: ⟨'f inference set⟩ and
  Bot-G :: ⟨'g set⟩ and
  Inf-G :: ⟨'g inference set⟩ and
  entails-G :: ⟨'g set ⇒ 'g set ⇒ bool⟩ (infix |=G 50) and
  Red-I-G :: ⟨'g set ⇒ 'g inference set⟩ and
  Red-F-G :: ⟨'g set ⇒ 'g set⟩
+ fixes
  Bot-F :: ⟨'f set⟩ and
  G-F :: ⟨'f ⇒ 'g set⟩ and
  G-I :: ⟨'f inference ⇒ 'g inference set option⟩
assumes
  Bot-F-not-empty: Bot-F ≠ {} and
  Bot-map-not-empty: ⟨B ∈ Bot-F ⇒ G-F B ≠ {}⟩ and
  Bot-map: ⟨B ∈ Bot-F ⇒ G-F B ⊆ Bot-G⟩ and

  strong-inf-map: ⟨ι ∈ Inf-F ⇒ G-I ι ≠ None ⇒ concl-of ' (the (G-I ι)) ⊆
(G-F (concl-of ι))⟩ and
  inf-map-in-Inf: ⟨ι ∈ Inf-F ⇒ G-I ι ≠ None ⇒ the (G-I ι) ⊆ Inf-G⟩
begin

sublocale light-standard-lifting Inf-F Bot-G Inf-G (|=G) Red-I-G Red-F-G Bot-F
G-F G-I
⟨proof⟩

end

```

7.3 Lifting with a Family of Tiebreaker Orderings

```

locale light-tiebreaker-lifting =
  empty-ord?: light-standard-lifting Inf-F Bot-G Inf-G entails-G Red-I-G Red-F-G
Bot-F G-F G-I
for
  Bot-F :: ⟨'f set⟩ and
  Inf-F :: ⟨'f inference set⟩ and
  Bot-G :: ⟨'g set⟩ and
  entails-G :: ⟨'g set ⇒ 'g set ⇒ bool⟩ (infix |=G 50) and
  Inf-G :: ⟨'g inference set⟩ and
  Red-I-G :: ⟨'g set ⇒ 'g inference set⟩ and
  Red-F-G :: ⟨'g set ⇒ 'g set⟩ and
  G-F :: 'f ⇒ 'g set and
  G-I :: 'f inference ⇒ 'g inference set option
+ fixes
  Prec-F-g :: ⟨'g ⇒ 'f ⇒ 'f ⇒ bool⟩
assumes
  all-wf: minimal-element (Prec-F-g g) UNIV
begin

```

definition $Red-F-\mathcal{G} :: 'f\ set \Rightarrow 'f\ set$ **where**

$\langle Red-F-\mathcal{G}\ N = \{C. \forall D \in \mathcal{G}-F\ C. D \in Red-F-G\ (\mathcal{G}-Fset\ N) \vee (\exists E \in N. Prec-F-g\ D\ E\ C \wedge D \in \mathcal{G}-F\ E)\} \rangle$

lemma *Prec-trans:*

assumes

$\langle Prec-F-g\ D\ A\ B \rangle$ **and**

$\langle Prec-F-g\ D\ B\ C \rangle$

shows

$\langle Prec-F-g\ D\ A\ C \rangle$

$\langle proof \rangle$

lemma *prop-nested-in-set:* $D \in P\ C \Longrightarrow C \in \{C. \forall D \in P\ C. A\ D \vee B\ C\ D\} \Longrightarrow A\ D \vee B\ C\ D$

$\langle proof \rangle$

lemma *Red-F-G-equiv-def:*

$\langle Red-F-\mathcal{G}\ N = \{C. \forall Di \in \mathcal{G}-F\ C. Di \in Red-F-G\ (\mathcal{G}-Fset\ N) \vee (\exists E \in (N - Red-F-\mathcal{G}\ N). Prec-F-g\ Di\ E\ C \wedge Di \in \mathcal{G}-F\ E)\} \rangle$

$\langle proof \rangle$

lemma *not-red-map-in-map-not-red:* $\langle \mathcal{G}-Fset\ N - Red-F-G\ (\mathcal{G}-Fset\ N) \subseteq \mathcal{G}-Fset\ (N - Red-F-\mathcal{G}\ N) \rangle$

$\langle proof \rangle$

lemma *Red-F-Bot-F:* $\langle B \in Bot-F \Longrightarrow N \models_{\mathcal{G}} \{B\} \Longrightarrow N - Red-F-\mathcal{G}\ N \models_{\mathcal{G}} \{B\} \rangle$

$\langle proof \rangle$

lemma *Red-F-of-subset-F:* $\langle N \subseteq N' \Longrightarrow Red-F-\mathcal{G}\ N \subseteq Red-F-\mathcal{G}\ N' \rangle$

$\langle proof \rangle$

lemma *Red-I-of-subset-F:* $\langle N \subseteq N' \Longrightarrow Red-I-\mathcal{G}\ N \subseteq Red-I-\mathcal{G}\ N' \rangle$

$\langle proof \rangle$

lemma *Red-F-of-Red-F-subset-F:* $\langle N' \subseteq Red-F-\mathcal{G}\ N \Longrightarrow Red-F-\mathcal{G}\ N \subseteq Red-F-\mathcal{G}\ (N - N') \rangle$

$\langle proof \rangle$

lemma *Red-I-of-Red-F-subset-F:* $\langle N' \subseteq Red-F-\mathcal{G}\ N \Longrightarrow Red-I-\mathcal{G}\ N \subseteq Red-I-\mathcal{G}\ (N - N') \rangle$

$\langle proof \rangle$

lemma *Red-I-of-Inf-to-N-F*:
assumes
i-in: $\langle \iota \in \text{Inf-F} \rangle$ **and**
concl-i-in: $\langle \text{concl-of } \iota \in N \rangle$
shows
 $\langle \iota \in \text{Red-I-}\mathcal{G} \ N \ \rangle$
 $\langle \text{proof} \rangle$

sublocale *calculus Bot-F Inf-F entails- $\mathcal{G}$ Red-I- $\mathcal{G}$ Red-F- $\mathcal{G}$*
 $\langle \text{proof} \rangle$

end

lemma *wf-empty-rel: minimal-element* $(\lambda - -. \text{False}) \text{ UNIV}$
 $\langle \text{proof} \rangle$

lemma *light-standard-empty-tiebreaker-equiv: light-standard-lifting Inf-F Bot-G Inf-G entails-G Red-I-G*
 $\text{Red-F-G Bot-F } \mathcal{G}\text{-F } \mathcal{G}\text{-I} = \text{light-tiebreaker-lifting Bot-F Inf-F Bot-G entails-G}$
 Inf-G Red-I-G
 $\text{Red-F-G } \mathcal{G}\text{-F } \mathcal{G}\text{-I } (\lambda g \ C \ C'. \text{False})$
 $\langle \text{proof} \rangle$

context *light-standard-lifting*
begin

interpretation *empt-ord: light-tiebreaker-lifting Bot-F Inf-F Bot-G entails-G*
 Inf-G Red-I-G
 $\text{Red-F-G } \mathcal{G}\text{-F } \mathcal{G}\text{-I } \lambda g \ C \ C'. \text{False}$
 $\langle \text{proof} \rangle$

lemma *red-f-equiv: empt-ord.Red-F- $\mathcal{G}$ = Red-F- $\mathcal{G}$*
 $\langle \text{proof} \rangle$

sublocale *calc?: calculus Bot-F Inf-F entails- $\mathcal{G}$ Red-I- $\mathcal{G}$ Red-F- $\mathcal{G}$*
 $\langle \text{proof} \rangle$

lemma *grounded-inf-in-ground-inf: $\iota \in \text{Inf-F} \implies \mathcal{G}\text{-I } \iota \neq \text{None} \implies \text{the } (\mathcal{G}\text{-I } \iota) \subseteq \text{Inf-G}$*
 $\langle \text{proof} \rangle$

abbreviation *ground-Inf-overapproximated* :: $'f \text{ set} \Rightarrow \text{bool}$ **where**
 $\text{ground-Inf-overapproximated } N \equiv \text{ground.Inf-from } (\mathcal{G}\text{-Fset } N)$
 $\subseteq \{ \iota. \exists \iota' \in \text{Inf-from } N. \mathcal{G}\text{-I } \iota' \neq \text{None} \wedge \iota \in \text{the } (\mathcal{G}\text{-I } \iota') \} \cup \text{Red-I-G } (\mathcal{G}\text{-Fset } N)$

lemma *sat-inf-imp-ground-red*:
assumes
 saturated N and
 $\iota' \in \text{Inf-from } N$ **and**
 $\mathcal{G}\text{-I } \iota' \neq \text{None} \wedge \iota \in \text{the } (\mathcal{G}\text{-I } \iota')$
shows $\iota \in \text{Red-I-G } (\mathcal{G}\text{-Fset } N)$
<proof>

lemma *sat-imp-ground-sat*:
saturated N $\implies$ ground-Inf-overapproximated N $\implies$ ground.saturated ($\mathcal{G}\text{-Fset } N$)
<proof>

end

context *light-tiebreaker-lifting*
begin

lemma *saturated-empty-order-equiv-saturated*:
saturated N = calc.saturated N
<proof>

lemma *static-empty-order-equiv-static*:
statically-complete-calculus Bot-F Inf-F entails- $\mathcal{G}$ Red-I- $\mathcal{G}$ Red-F- $\mathcal{G}$ =
statically-complete-calculus Bot-F Inf-F entails- $\mathcal{G}$ Red-I- $\mathcal{G}$ empty-ord.Red-F- $\mathcal{G}$
<proof>

theorem *static-to-dynamic*:
statically-complete-calculus Bot-F Inf-F entails- $\mathcal{G}$ Red-I- $\mathcal{G}$ empty-ord.Red-F- $\mathcal{G}$ =
dynamically-complete-calculus Bot-F Inf-F entails- $\mathcal{G}$ Red-I- $\mathcal{G}$ Red-F- $\mathcal{G}$
<proof>

end

7.4 Lifting with a Family of Redundancy Criteria

locale *light-lifting-intersection* = *inference-system Inf-F +*
ground: inference-system-family Q Inf-G-q +
ground: consequence-relation-family Bot-G Q entails-q
for
Inf-F :: 'f inference set and

$Bot-G :: 'g \text{ set and}$
 $Q :: 'q \text{ set and}$
 $Inf-G-q :: \langle 'q \Rightarrow 'g \text{ inference set} \rangle \text{ and}$
 $entails-q :: 'q \Rightarrow 'g \text{ set} \Rightarrow 'g \text{ set} \Rightarrow \text{bool and}$
 $Red-I-q :: 'q \Rightarrow 'g \text{ set} \Rightarrow 'g \text{ inference set and}$
 $Red-F-q :: 'q \Rightarrow 'g \text{ set} \Rightarrow 'g \text{ set}$
+ **fixes**
 $Bot-F :: 'f \text{ set and}$
 $\mathcal{G}-F-q :: 'q \Rightarrow 'f \Rightarrow 'g \text{ set and}$
 $\mathcal{G}-I-q :: 'q \Rightarrow 'f \text{ inference} \Rightarrow 'g \text{ inference set option and}$
 $Prec-F-g :: 'g \Rightarrow 'f \Rightarrow 'f \Rightarrow \text{bool}$
assumes
light-standard-lifting-family:
 $\forall q \in Q. \text{light-tiebreaker-lifting } Bot-F \text{ Inf-F } Bot-G \text{ (entails-q } q) \text{ (Inf-G-q } q)$
 $(Red-I-q \text{ } q)$
 $(Red-F-q \text{ } q) (\mathcal{G}-F-q \text{ } q) (\mathcal{G}-I-q \text{ } q) \text{ Prec-F-g}$
begin

abbreviation $\mathcal{G}-Fset-q :: 'q \Rightarrow 'f \text{ set} \Rightarrow 'g \text{ set where}$
 $\mathcal{G}-Fset-q \text{ } q \text{ } N \equiv \bigcup (\mathcal{G}-F-q \text{ } q \text{ ' } N)$

definition $Red-I-\mathcal{G}-q :: 'q \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ inference set where}$
 $Red-I-\mathcal{G}-q \text{ } q \text{ } N = \{\iota \in Inf-F. (\mathcal{G}-I-q \text{ } q \text{ } \iota \neq None \wedge \text{the } (\mathcal{G}-I-q \text{ } q \text{ } \iota) \subseteq Red-I-q \text{ } q$
 $(\mathcal{G}-Fset-q \text{ } q \text{ } N))$
 $\vee (\mathcal{G}-I-q \text{ } q \text{ } \iota = None \wedge \mathcal{G}-F-q \text{ } q \text{ (concl-of } \iota) \subseteq (\mathcal{G}-Fset-q \text{ } q \text{ } N \cup Red-F-q \text{ } q$
 $(\mathcal{G}-Fset-q \text{ } q \text{ } N)))\}$

definition $Red-F-\mathcal{G}-empty-q :: 'q \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ set where}$
 $Red-F-\mathcal{G}-empty-q \text{ } q \text{ } N = \{C. \forall D \in \mathcal{G}-F-q \text{ } q \text{ } C. D \in Red-F-q \text{ } q (\mathcal{G}-Fset-q \text{ } q \text{ } N)\}$

definition $Red-F-\mathcal{G}-q :: 'q \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ set where}$
 $Red-F-\mathcal{G}-q \text{ } q \text{ } N =$
 $\{C. \forall D \in \mathcal{G}-F-q \text{ } q \text{ } C. D \in Red-F-q \text{ } q (\mathcal{G}-Fset-q \text{ } q \text{ } N) \vee (\exists E \in N. Prec-F-g \text{ } D$
 $E \text{ } C \wedge D \in \mathcal{G}-F-q \text{ } q \text{ } E)\}$

abbreviation $entails-\mathcal{G}-q :: 'q \Rightarrow 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow \text{bool where}$
 $entails-\mathcal{G}-q \text{ } q \text{ } N1 \text{ } N2 \equiv entails-q \text{ } q (\mathcal{G}-Fset-q \text{ } q \text{ } N1) (\mathcal{G}-Fset-q \text{ } q \text{ } N2)$

lemma *red-crit-lifting-family:*
assumes $q\text{-in}: q \in Q$
shows *calculus* $Bot-F \text{ Inf-F (entails-}\mathcal{G}\text{-q } q) (Red-I-\mathcal{G}\text{-q } q) (Red-F-\mathcal{G}\text{-q } q)$
 $\langle \text{proof} \rangle$

lemma *red-crit-lifting-family-empty-ord:*
assumes $q\text{-in}: q \in Q$
shows *calculus* $Bot-F \text{ Inf-F (entails-}\mathcal{G}\text{-q } q) (Red-I-\mathcal{G}\text{-q } q) (Red-F-\mathcal{G}\text{-empty-q } q)$
 $\langle \text{proof} \rangle$

sublocale *consequence-relation-family* $Bot-F \text{ } Q \text{ entails-}\mathcal{G}\text{-q}$

$\langle \text{proof} \rangle$

sublocale *intersection-calculus Bot-F Inf-F Q entails- $\mathcal{G}$ -q Red-I- $\mathcal{G}$ -q Red-F- $\mathcal{G}$ -q*
 $\langle \text{proof} \rangle$

abbreviation *entails- $\mathcal{G}$:: 'f set $\Rightarrow$ 'f set $\Rightarrow$ bool (infix $\models_{\cap \mathcal{G}}$ 50) where*
 $(\models_{\cap \mathcal{G}}) \equiv \text{entails}$

abbreviation *Red-I- $\mathcal{G}$:: 'f set $\Rightarrow$ 'f inference set where*
 $\text{Red-I-}\mathcal{G} \equiv \text{Red-I}$

abbreviation *Red-F- $\mathcal{G}$:: 'f set $\Rightarrow$ 'f set where*
 $\text{Red-F-}\mathcal{G} \equiv \text{Red-F}$

lemmas *entails- $\mathcal{G}$ -def = entails-def*

lemmas *Red-I- $\mathcal{G}$ -def = Red-I-def*

lemmas *Red-F- $\mathcal{G}$ -def = Red-F-def*

sublocale *empty-ord: intersection-calculus Bot-F Inf-F Q entails- $\mathcal{G}$ -q Red-I- $\mathcal{G}$ -q*
Red-F- $\mathcal{G}$ -empty-q
 $\langle \text{proof} \rangle$

abbreviation *Red-F- $\mathcal{G}$ -empty :: 'f set $\Rightarrow$ 'f set where*
 $\text{Red-F-}\mathcal{G}\text{-empty} \equiv \text{empty-ord.Red-F}$

lemmas *Red-F- $\mathcal{G}$ -empty-def = empty-ord.Red-F-def*

lemma *sat-inf-imp-ground-red-fam-inter:*

assumes

sat-n: saturated N and

i'-in: $\iota' \in \text{Inf-from } N$ and

q-in: $q \in Q$ and

grounding: $\mathcal{G}\text{-I-}q \ q \ \iota' \neq \text{None} \wedge \iota \in \text{the } (\mathcal{G}\text{-I-}q \ q \ \iota')$

shows *$\iota \in \text{Red-I-}q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N)$*

$\langle \text{proof} \rangle$

abbreviation *ground-Inf-overapproximated :: 'q $\Rightarrow$ 'f set $\Rightarrow$ bool where*

ground-Inf-overapproximated $q \ N \equiv$

ground.Inf-from- $q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N)$

$\subseteq \{\iota. \exists \iota' \in \text{Inf-from } N. \mathcal{G}\text{-I-}q \ q \ \iota' \neq \text{None} \wedge \iota \in \text{the } (\mathcal{G}\text{-I-}q \ q \ \iota')\} \cup \text{Red-I-}q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N)$

abbreviation *ground-saturated :: 'q $\Rightarrow$ 'f set $\Rightarrow$ bool where*

ground-saturated $q \ N \equiv \text{ground.Inf-from-}q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N) \subseteq \text{Red-I-}q \ q \ (\mathcal{G}\text{-Fset-}q \ q \ N)$

lemma *sat-imp-ground-sat-fam-inter:*

saturated $N \implies q \in Q \implies \text{ground-Inf-overapproximated } q \ N \implies \text{ground-saturated } q \ N$

<proof>

lemma *sat-eq-sat-empty-order*: *saturated N = empty-ord.saturated N*
<proof>

lemma *static-empty-ord-inter-equiv-static-inter*:
statically-complete-calculus Bot-F Inf-F entails Red-I Red-F =
statically-complete-calculus Bot-F Inf-F entails Red-I Red-F- $\mathcal{G}$ -empty
<proof>

theorem *stat-eq-dyn-ref-comp-fam-inter*: *statically-complete-calculus Bot-F Inf-F*
entails Red-I Red-F- $\mathcal{G}$ -empty =
dynamically-complete-calculus Bot-F Inf-F entails Red-I Red-F
<proof>

end

end

theory *List-Extra*

imports *Main HOL-Library.Quotient-List*

begin

This theory contains some extra lemmas that were useful in proving some lemmas in `Modular_Splitting_Calculus.thy` and `Lightweight_Avatar.thy`.

lemma *map2-first-is-first* [simp]: *<length x = length y $\implies$ map2 (λ x y. x) x y = x>*
<proof>

lemma *map2-second-is-second* [simp]: *<length A = length B $\implies$ map2 (λ x y. y) A B = B>*
<proof>

lemma *list-all-exists-is-exists-list-all2*:
assumes *<list-all (λ x. $\exists$ y. P x y) xs>*
shows *< $\exists$ ys. list-all2 P xs ys>*
<proof>

lemma *ball-set-f-to-ball-set-map*: *<($\forall$ x $\in$ set A. P (f x)) $\longleftrightarrow$ ($\forall$ x $\in$ set (map f A). P x)>*
<proof>

lemma *list-all-ex-to-ex-list-all2*:
<list-all (λ x. $\exists$ y. P x y) A $\longleftrightarrow$ ($\exists$ ys. length A = length ys $\wedge$ list-all2 (λ x y.

$P\ x\ y)\ A\ ys\rangle$
 $\langle proof\rangle$

lemma *list-all2-to-map*:

assumes *lengths-eq*: $\langle length\ A = length\ B\rangle$
shows $\langle list-all2\ (\lambda\ x\ y.\ P\ (f\ x\ y))\ A\ B \longleftrightarrow list-all\ P\ (map2\ f\ A\ B)\rangle$
 $\langle proof\rangle\langle proof\rangle\langle proof\rangle\langle proof\rangle\langle proof\rangle\langle proof\rangle\langle proof\rangle\langle proof\rangle\langle proof\rangle\langle proof\rangle$
end

theory *Modular-Splitting-Calculus*

imports

Calculi-And-Annotations

Light-Lifting-to-Non-Ground-Calculi

List-Extra

FSet-Extra

begin

8 Core splitting calculus

In this section, we formalize an abstract version of a splitting calculus. We start by considering only two basic rules:

- BASE performs an inference from our inference system;
- UNSAT replaces a set of propositionally unsatisfiable formulas with $\perp$.

locale *annotated-calculus* = *calculus-with-annotated-consrel* **bot** *FInf* *entails* *entails-sound* *FRed_I*

FRed_F *fml* *asn*

for *bot* :: '*f*' **and**

FInf :: $\langle 'f\ inference\ set\rangle$ **and**

entails :: $\langle ['f\ set,\ 'f\ set] \Rightarrow bool\rangle$ (**infix** $\langle \models \rangle$ 50) **and**

entails-sound :: $\langle ['f\ set,\ 'f\ set] \Rightarrow bool\rangle$ (**infix** $\langle \models_s \rangle$ 50) **and**

FRed_I :: $\langle 'f\ set \Rightarrow 'f\ inference\ set\rangle$ **and**

FRed_F :: $\langle 'f\ set \Rightarrow 'f\ set\rangle$ **and**

fml :: $\langle 'v :: countable \Rightarrow 'f\rangle$ **and**

asn :: $\langle 'f\ sign \Rightarrow 'v\ sign\ set\rangle$

+ assumes

entails-nontrivial: $\langle \neg\ \{\}\ \models\ \{\}\rangle$ **and**

reducedness: $\langle Inf-between\ UNIV\ (FRed_F\ N) \subseteq FRed_I\ N\rangle$ **and**

complete: $\langle bot \notin FRed_F\ N\rangle$ **and**

all-red-to-bot: $\langle C \neq bot \implies C \in FRed_F\ \{bot\}\rangle$

begin

notation *sound-cons.entails-neg* (**infix** $\langle \models_{s\sim} \rangle$ 50)

8.1 The inference rules

Every inference rule X is defined using two functions: X_pre and X_inf . X_inf is the inference rule itself, while X_pre are side-conditions for the rule to be applicable.

abbreviation *base-pre* :: $\langle ('f, 'v) \text{ AF list} \Rightarrow 'f \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{base-pre } \mathcal{N} \ D \equiv \text{Infer } (\text{map } F\text{-of } \mathcal{N}) \ D \in F\text{Inf} \rangle$

abbreviation *base-inf* :: $\langle ('f, 'v) \text{ AF list} \Rightarrow 'f \Rightarrow ('f, 'v) \text{ AF inference} \rangle$ **where**
 $\langle \text{base-inf } \mathcal{N} \ D \equiv \text{Infer } \mathcal{N} \ (\text{Pair } D \ (\text{ffUnion } (\text{fset-of-list } (\text{map } A\text{-of } \mathcal{N})))) \rangle$

abbreviation *unsat-pre* :: $\langle ('f, 'v) \text{ AF list} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{unsat-pre } \mathcal{N} \equiv (\forall x \in \text{set } \mathcal{N}. F\text{-of } x = \text{bot}) \wedge \text{propositionally-unsatisfiable } (\text{set } \mathcal{N}) \rangle$

abbreviation *unsat-inf* :: $\langle ('f, 'v) \text{ AF list} \Rightarrow ('f, 'v) \text{ AF inference} \rangle$ **where**
 $\langle \text{unsat-inf } \mathcal{N} \equiv \text{Infer } \mathcal{N} \ (\text{to-AF bot}) \rangle$

We consider first only the inference rules **BASE** and **UNSAT**. The optional inference and simplification rules are handled separately in the locales *splitting-calculus-extensions* and *splitting-calculus-with-simps* respectively.

inductive-set *SInf* :: $\langle ('f, 'v) \text{ AF inference set} \rangle$ **where**
 $\text{base: } \langle \text{base-pre } \mathcal{N} \ D \Longrightarrow \text{base-inf } \mathcal{N} \ D \in S\text{Inf} \rangle$
 $\mid \text{unsat: } \langle \text{unsat-pre } \mathcal{N} \Longrightarrow \text{unsat-inf } \mathcal{N} \in S\text{Inf} \rangle$

The predicates in *Splitting-rules* form a valid inference system.

interpretation *SInf-inf-system*: *inference-system* *SInf* $\langle \text{proof} \rangle$

lemma *not-empty-entails-bot*: $\langle \neg\{\} \models \{\text{bot}\} \rangle$
 $\langle \text{proof} \rangle$

The proof for Lemma 13 is split into two parts, for each inclusion in the set equality.

lemma *SInf-commutes-Inf1*:
 $\langle \text{bot} \notin \mathcal{N} \ \text{proj}_J \ J \Longrightarrow (\text{inference-system.} \text{Inf-from } S\text{Inf } \mathcal{N}) \ \iota_{\text{proj}_J} \ J \subseteq \text{Inf-from } (\mathcal{N} \ \text{proj}_J \ J) \rangle$
 $\langle \text{proof} \rangle$

lemma *SInf-commutes-Inf2*:
 $\langle \text{bot} \notin \mathcal{N} \ \text{proj}_J \ J \Longrightarrow \text{Inf-from } (\mathcal{N} \ \text{proj}_J \ J) \subseteq (\text{inference-system.} \text{Inf-from } S\text{Inf } \mathcal{N}) \ \iota_{\text{proj}_J} \ J \rangle$
 $\langle \text{proof} \rangle$

We use $\text{bot} \notin ?\mathcal{N} \text{ proj}_J ?J \implies \text{SInf-inf-system.Inf-from } ?\mathcal{N} \iota\text{proj}_J ?J \subseteq \text{Inf-from } (? \mathcal{N} \text{ proj}_J ?J)$ and $\text{bot} \notin ?\mathcal{N} \text{ proj}_J ?J \implies \text{Inf-from } (? \mathcal{N} \text{ proj}_J ?J) \subseteq \text{SInf-inf-system.Inf-from } ?\mathcal{N} \iota\text{proj}_J ?J$ to put the Lemma 13 together into a single proof.

lemma *SInf-commutes-Inf*:

$\langle \text{bot} \notin \mathcal{N} \text{ proj}_J J \implies (\text{inference-system.Inf-from SInf } \mathcal{N}) \iota\text{proj}_J J = \text{Inf-from } (\mathcal{N} \text{ proj}_J J) \rangle$
 $\langle \text{proof} \rangle$

theorem *SInf-sound-wrt-entails-sound*: $\langle \iota_S \in \text{SInf} \implies \text{set } (\text{prems-of } \iota_S) \models_{\text{SAF}} \{\text{concl-of } \iota_S\} \rangle$
 $\langle \text{proof} \rangle$

The lifted calculus provides a consequence relation and a sound inference system.

interpretation *AF-sound-cons-rel*: *consequence-relation* $\langle \text{to-AF bot} \rangle \langle \models_{\text{SAF}} \rangle$
 $\langle \text{proof} \rangle$

interpretation *SInf-sound-inf-system*: *sound-inference-system* *SInf* $\langle \text{to-AF bot} \rangle \langle \models_{\text{SAF}} \rangle$
 $\langle \text{proof} \rangle$

8.2 The redundancy criterion

definition *SRed_F* :: $\langle ('f, 'v) \text{ AF set} \implies ('f, 'v) \text{ AF set} \rangle$ **where**

$\langle \text{SRed}_F \mathcal{N} = \{ \text{AF.Pair } C A \mid C A. \forall \mathcal{J}. \text{total-strip } \mathcal{J} \supseteq \text{fset } A \longrightarrow C \in \text{FRed}_F (\mathcal{N} \text{ proj}_J \mathcal{J}) \} \cup \{ \text{AF.Pair } C A \mid C A. \exists \mathcal{C} \in \mathcal{N}. \text{F-of } \mathcal{C} = C \wedge \text{A-of } \mathcal{C} \sqsubset A \} \rangle$

definition *SRed_I* :: $\langle ('f, 'v) \text{ AF set} \implies ('f, 'v) \text{ AF inference set} \rangle$ **where**

$\langle \text{SRed}_I \mathcal{N} = \{ \text{base-inf } \mathcal{M} C \mid \mathcal{M} C. \text{base-pre } \mathcal{M} C \wedge (\forall \mathcal{J}. \{ \text{base-inf } \mathcal{M} C \} \iota\text{proj}_J \mathcal{J} \subseteq \text{FRed}_I (\mathcal{N} \text{ proj}_J \mathcal{J})) \} \cup \{ \text{unsat-inf } \mathcal{M} \mid \mathcal{M}. \text{unsat-pre } \mathcal{M} \wedge \text{to-AF bot} \in \mathcal{N} \} \rangle$

lemma *sredI-N-proj-J-subset-redI-proj-J*: $\langle \text{to-AF bot} \notin \mathcal{N} \implies (\text{SRed}_I \mathcal{N}) \iota\text{proj}_J J \subseteq \text{FRed}_I (\mathcal{N} \text{ proj}_J J) \rangle$
 $\langle \text{proof} \rangle$

lemma *bot-not-in-sredF-N*: $\langle \text{to-AF bot} \notin \text{SRed}_F \mathcal{N} \rangle$
 $\langle \text{proof} \rangle$

We need to set things up for the proof of lemma 18. We first restrict *SRed_I* to BASE inferences (under the name *ARed_I*) and show that it is a redundancy criterion. And then we consider the case of UNSAT inferences separately.

definition $ARed_F :: \langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \rangle$ **where**
 $\langle ARed_F \mathcal{N} \equiv SRed_F \mathcal{N} \rangle$

definition $ARed_I :: \langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF inference set} \rangle$ **where**
 $\langle ARed_I \mathcal{N} \equiv \{ \text{base-inf } \mathcal{M} \mathcal{C} \mid \mathcal{M} \mathcal{C}. \text{base-pre } \mathcal{M} \mathcal{C} \wedge$
 $(\forall \mathcal{J}. \{ \text{base-inf } \mathcal{M} \mathcal{C} \} \iota \text{proj}_J \mathcal{J} \subseteq FRed_I (\mathcal{N} \text{proj}_J \mathcal{J})) \} \rangle$

definition $AInf :: \langle ('f, 'v) \text{ AF inference set} \rangle$ **where**
 $\langle AInf \equiv \{ \text{base-inf } \mathcal{N} D \mid \mathcal{N} D. \text{base-pre } \mathcal{N} D \} \rangle$

definition $\mathcal{G}_F :: \langle 'v \text{ total-interpretation} \Rightarrow ('f, 'v) \text{ AF} \Rightarrow 'f \text{ set} \rangle$ **where**
 $\langle \mathcal{G}_F \mathcal{J} \mathcal{C} \equiv \{ \mathcal{C} \} \text{proj}_J \mathcal{J} \rangle$

definition $\mathcal{G}_I :: \langle 'v \text{ total-interpretation} \Rightarrow ('f, 'v) \text{ AF inference} \Rightarrow 'f \text{ inference set} \rangle$
where
 $\langle \mathcal{G}_I \mathcal{J} \iota \equiv \{ \iota \} \iota \text{proj}_J \mathcal{J} \rangle$

We define a wellfounded ordering on A-formulas to strengthen $ARed_I$.
Basically, $A \leftarrow \mathcal{C} \sqsupset A \leftarrow \mathcal{C}'$ if $\mathcal{C} \subset \mathcal{C}'$.

definition $\text{tiebreaker-order} :: \langle ('f, 'v :: \text{countable}) \text{ AF rel} \rangle$ **where**
 $\langle \text{tiebreaker-order} \equiv \{ (\mathcal{C}, \mathcal{C}'). F\text{-of } \mathcal{C} = F\text{-of } \mathcal{C}' \wedge A\text{-of } \mathcal{C} \mid \sqsubset A\text{-of } \mathcal{C}' \} \rangle$

abbreviation $\text{sqsupset-is-tiebreaker-order}$ (**infix** $\langle \sqsupset \rangle$ 50) **where**
 $\langle \mathcal{C} \sqsupset \mathcal{C}' \equiv (\mathcal{C}, \mathcal{C}') \in \text{tiebreaker-order} \rangle$

lemma $\text{tiebreaker-order-is-strict-partial-order}$: $\langle \text{po-on } (\sqsupset) \text{ UNIV} \rangle$
 $\langle \text{proof} \rangle$

lemma wfp-on-fsubset : $\langle \text{wfp-on } (\mid \sqsubset) \text{ UNIV} \rangle$
 $\langle \text{proof} \rangle$

lemma $\text{wfp-on-tiebreaker-order}$: $\langle \text{wfp-on } (\sqsupset) (\text{UNIV} :: ('f, 'v) \text{ AF set}) \rangle$
 $\langle \text{proof} \rangle$

We can lift inferences from $FRed_I$ to $ARed_I$.

interpretation $\text{lift-from-FRed-to-ARed}$: $\text{light-tiebreaker-lifting } \langle \{ \text{to-AF bot} \} \rangle AInf$
 $\langle \{ \text{bot} \} \rangle \langle (\models \cap) \rangle$
 $FInf FRed_I FRed_F \langle \mathcal{G}_F \mathcal{J} \rangle \langle \text{Some} \circ \mathcal{G}_I \mathcal{J} \rangle \langle \lambda \cdot. (\sqsupset) \rangle$
 $\langle \text{proof} \rangle$

lemma $ARed_I\text{-is-FRed}_I$: $\langle ARed_I \mathcal{N} = (\bigcap J. \text{lift-from-FRed-to-ARed.Red-I-}\mathcal{G} J \mathcal{N}) \rangle$
 $\langle \text{proof} \rangle$

lemma $ARed_F\text{-is-FRed}_F$: $\langle ARed_F \mathcal{N} = (\bigcap J. \text{lift-from-FRed-to-ARed.Red-F-}\mathcal{G} J \mathcal{N}) \rangle$
 $\langle \text{proof} \rangle$

lemma *entails-is-entails-G*: $\langle \mathcal{M} \models_{AF} \{C\} \longleftrightarrow (\forall \mathcal{J}. \text{lift-from-FRed-to-ARed.entails-G } \mathcal{J} \mathcal{M} \{C\}) \rangle$
 $\langle \text{proof} \rangle$

lemma *SRed_I-in-SInf*: $\langle SRed_I N \subseteq SInf \rangle$
 $\langle \text{proof} \rangle$

lemma *SRed_F-entails-bot*: $\langle N \models_{AF} \{to\text{-}AF \text{ bot} \} \implies N - SRed_F N \models_{AF} \{to\text{-}AF \text{ bot} \} \rangle$
 $\langle \text{proof} \rangle$

lemma *SRed_F-of-subset-F*: $\langle N \subseteq N' \implies SRed_F N \subseteq SRed_F N' \rangle$
 $\langle \text{proof} \rangle$

lemma *SRed_I-of-subset-F*: $\langle N \subseteq N' \implies SRed_I N \subseteq SRed_I N' \rangle$
 $\langle \text{proof} \rangle$

lemma *SRed_F-of-SRed_F-subset-F*: $\langle N' \subseteq SRed_F N \implies SRed_F N \subseteq SRed_F (N - N') \rangle$
 $\langle \text{proof} \rangle$

lemma *SRed_I-of-SRed_F-subset-F*: $\langle N' \subseteq SRed_F N \implies SRed_I N \subseteq SRed_I (N - N') \rangle$
 $\langle \text{proof} \rangle$

lemma *SRed_I-of-SInf-to-N-F*: $\langle \iota_S \in SInf \implies \text{concl-of } \iota_S \in N \implies \iota_S \in SRed_I N \rangle$
 $\langle \text{proof} \rangle$

end

8.3 Standard completeness

context *annotated-calculus*
begin

lemmas *SRed-rules* = *SRed_F-entails-bot* *SRed_F-of-subset-F* *SRed_I-of-subset-F* *SRed_F-of-SRed_F-subset-F*
SRed_I-of-SRed_F-subset-F *SRed_I-of-SInf-to-N-F* *SRed_I-in-SInf*

sublocale *S-calculus*: *calculus* $\langle to\text{-}AF \text{ bot} \rangle$ *SInf* *AF-entails* *SRed_I* *SRed_F*
 $\langle \text{proof} \rangle$

lemma *S-saturated-to-F-saturated*: $\langle S\text{-calculus.saturated } \mathcal{N} \implies \text{saturated } (\mathcal{N} \text{ proj}_J) \rangle$

$\mathcal{J}\rangle$
 $\langle proof \rangle$

notation *AF-cons-rel.entails-conjunctive* (**infix** $\langle \models_{AF} \rangle$ 50)

theorem *S-calculus-statically-complete:*

assumes *F-statically-complete:* $\langle statically-complete-calculus \ bot \ FInf \ (\models) \ FRed_I \ FRed_F \rangle$

shows $\langle statically-complete-calculus \ (to-AF \ bot) \ SInf \ (\models_{AF}) \ SRed_I \ SRed_F \rangle$
 $\langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle$

The following proof works as follows.

We assume that $(Inf, (Red_I, Red_F))$ is statically complete. From that and theorem *Calculi-And-Annotations.statically-complete-calculus bot FInf* $(\models) FRed_I FRed_F \implies Calculi-And-Annotations.statically-complete-calculus (to-AF \ bot) SInf (\models_{AF}) SRed_I SRed_F$, we obtain that $(SInf, (SRed_I, SRed_F))$ is statically complete. This means that for all $\mathcal{N} \subseteq UNIV$, if $\mathcal{N}$ is saturated w.r.t. $(SInf, SRed_I)$ and $\mathcal{N} \models_{\cup_{AF}} \{\perp\}$ then $\perp \in \mathcal{N}$. Since $\models_{\cup_{AF}} \equiv \models_{\cap_{AF}}$ when the right hand side is a singleton set, we have that for all $\mathcal{N} \subseteq UNIV$, if $\mathcal{N}$ is saturated w.r.t. $(SInf, SRed_I)$ and $\mathcal{N} \models_{\cap_{AF}} \{\perp\}$ then $\perp \in \mathcal{N}$.

Because $\models_{\cap_{AF}}$ is a consequence relation for the Saturation Framework, we can derive that $(SInf, (SRed_I, SRed_F))$ is dynamically complete (using the conjunctive entailment). We then proceed as above but in the opposite way to show that $(SInf, (SRed_I, SRed_F))$ is dynamically complete using the disjunctive entailment $\models_{\cup_{AF}}$.

corollary *S-calculus-dynamically-complete:*

assumes *F-statically-complete:* $\langle statically-complete-calculus \ bot \ FInf \ (\models) \ FRed_I \ FRed_F \rangle$

shows $\langle dynamically-complete-calculus \ (to-AF \ bot) \ SInf \ (\models_{AF}) \ SRed_I \ SRed_F \rangle$
 $\langle proof \rangle$

8.4 Strong completeness

theorem *S-calculus-strong-statically-complete:*

assumes *F-statically-complete:* $\langle statically-complete-calculus \ bot \ FInf \ (\models) \ FRed_I \ FRed_F \rangle$ **and**

$\mathcal{N}$ -*locally-saturated:* $\langle locally-saturated \ \mathcal{N} \rangle$ **and**

$\mathcal{N}$ -*entails-bot:* $\langle \mathcal{N} \models_{AF} \{to-AF \ bot\} \rangle$

shows $\langle to-AF \ bot \in \mathcal{N} \rangle$

$\langle proof \rangle \langle proof \rangle \langle proof \rangle \langle proof \rangle$

lemma *locally-fair-derivation-is-saturated-at-liminf:*

$\langle is-derivation \ S-calculus.derive \ \mathcal{N} \ i \implies locally-fair \ \mathcal{N} \ i \implies locally-saturated \ (lim-inf \ \mathcal{N} \ i) \rangle$

$\langle proof \rangle \langle proof \rangle$

theorem *S-calculus-strong-dynamically-complete:*

assumes *F-statically-complete:* $\langle statically-complete-calculus\ bot\ FInf\ (\models)\ FRed_I\ FRed_F \rangle$ **and**

Ni-is-derivation: $\langle is-derivation\ S-calculus.derive\ Ni \rangle$ **and**

Ni-is-locally-fair: $\langle locally-fair\ Ni \rangle$ **and**

Ni0-entails-bot: $\langle llhd\ Ni\ \models_{AF}\ \{to-AF\ bot\} \rangle$

shows $\langle \exists\ i.\ to-AF\ bot \in llnth\ Ni\ i \rangle$

$\langle proof \rangle$

end

9 Extensions: Inferences and simplifications

9.1 Simplifications

datatype *'f simplification* =

Simplify (*S-from:* $\langle 'f\ set \rangle$) (*S-to:* $\langle 'f\ set \rangle$)

Simplification rules are said to be sound if every conclusion is entailed by all premises. We could have also used our conjunctive entailment *consequence-relation.entails-conjunctive*, but it is defined that way so there is nothing to worry about.

locale *AF-calculus* = *sc:* *sound-calculus bot Inf entails entails-sound Red_I Red_F*
for

bot :: $\langle 'f, 'v :: countable \rangle AF$ **and**

Inf :: $\langle 'f, 'v \rangle AF\ inference\ set$ **and**

entails :: $\langle 'f, 'v \rangle AF\ set \Rightarrow \langle 'f, 'v \rangle AF\ set \Rightarrow bool$ **and**

entails-sound :: $\langle 'f, 'v \rangle AF\ set \Rightarrow \langle 'f, 'v \rangle AF\ set \Rightarrow bool$ **and**

Red_I :: $\langle 'f, 'v \rangle AF\ set \Rightarrow \langle 'f, 'v \rangle AF\ inference\ set$ **and**

Red_F :: $\langle 'f, 'v \rangle AF\ set \Rightarrow \langle 'f, 'v \rangle AF\ set$

locale *AF-calculus-extended* =

AF-calculus bot Inf entails entails-sound Red_I Red_F

for *bot* :: $\langle 'f, 'v :: countable \rangle AF$ **and**

Inf :: $\langle 'f, 'v \rangle AF\ inference\ set$ **and**

entails :: $\langle 'f, 'v \rangle AF\ set \Rightarrow \langle 'f, 'v \rangle AF\ set \Rightarrow bool$ **and**

entails-sound :: $\langle 'f, 'v \rangle AF\ set \Rightarrow \langle 'f, 'v \rangle AF\ set \Rightarrow bool$ (**infix** $\langle \models_s \rangle\ 50$)

and

Red_I :: $\langle 'f, 'v \rangle AF\ set \Rightarrow \langle 'f, 'v \rangle AF\ inference\ set$ **and**

Red_F :: $\langle 'f, 'v \rangle AF\ set \Rightarrow \langle 'f, 'v \rangle AF\ set$

+ fixes

Simps :: $\langle 'f, 'v \rangle AF\ simplification\ set$ **and**

OptInfs :: $\langle 'f, 'v \rangle AF\ inference\ set$

assumes

simps-simp: $\langle \delta \in Simps \Rightarrow (S-from\ \delta - S-to\ \delta) \subseteq Red_F\ (S-to\ \delta) \rangle$ **and**

simps-sound: $\langle \delta \in \text{Simps} \implies \forall C \in S\text{-to } \delta. S\text{-from } \delta \models_s \{C\} \rangle$ **and**
infs-sound: $\langle \iota \in \text{OptInfs} \implies \text{set } (\text{prems-of } \iota) \models_s \{\text{concl-of } \iota\} \rangle$
begin
lemma *simp-in-derivations*: $\langle \delta \in \text{Simps} \implies$
sc.derive $(\mathcal{M} \cup S\text{-from } \delta) (\mathcal{M} \cup S\text{-to } \delta) \rangle$
 $\langle \text{proof} \rangle$
lemma *opt-infs-in-derivations*: $\langle \iota \in \text{OptInfs} \implies$
sc.derive $(\mathcal{M} \cup \text{set } (\text{prems-of } \iota)) (\mathcal{M} \cup \text{set } (\text{prems-of } \iota) \cup \{\text{concl-of } \iota\}) \rangle$
 $\langle \text{proof} \rangle$
end

Empty sets of simplifications and optional inferences are accepted in termlocale *AF-calculus-extended*

context *AF-calculus*
begin

sublocale *empty-simps*:
AF-calculus-extended bot Inf entails entails-sound Red_I Red_F
 $\{\} :: (f, v) \text{ AF simplification set } \{\} :: (f, v) \text{ AF inference set}$
 $\langle \text{proof} \rangle$

end

Here we extend our basic calculus with simplification rules, one at a time:

- SPLIT performs a n -ary case analysis on the head of the premise;
- COLLECT performs garbage collection on clauses which contain propositionally unsatisfiable heads;
- TRIM removes assertions which are entailed by others.

9.1.1 The Split Rule

locale *AF-calculus-with-split* =
base-calculus: *AF-calculus-extended bot SInf entails entails-sound SRed_I*
SRed_F Simps OptInfs
for *bot* :: $\langle (f, v :: \text{countable}) \text{ AF} \rangle$ **and**
SInf :: $\langle (f, v) \text{ AF inference set} \rangle$ **and**
entails :: $\langle (f, v) \text{ AF set} \Rightarrow (f, v) \text{ AF set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{AF} \rangle$ 50) **and**
entails-sound :: $\langle (f, v) \text{ AF set} \Rightarrow (f, v) \text{ AF set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{AFS} \rangle$ 50)
and
SRed_I :: $\langle (f, v) \text{ AF set} \Rightarrow (f, v) \text{ AF inference set} \rangle$ **and**
SRed_F :: $\langle (f, v) \text{ AF set} \Rightarrow (f, v) \text{ AF set} \rangle$ **and**

$Simps :: \langle ('f, 'v) \text{ AF simplification set} \rangle$ **and**
 $OptInfs :: \langle ('f, 'v) \text{ AF inference set} \rangle$
+ fixes
 $splittable :: \langle ('f, 'v) \text{ AF} \Rightarrow ('f, 'v) \text{ AF fset} \Rightarrow \text{bool} \rangle$
assumes
 $split\text{-}sound1: \langle splittable \ C \ Cs \Longrightarrow$
 $\{C\} \models_{AFS} \{AF.Pair \ (F\text{-of} \ bot) \ (ffUnion \ (fimage \ neg \ |\cdot| \ A\text{-of} \ |\cdot| \ Cs) \ |\cup| \ A\text{-of} \ C)\} \rangle$ **and**
 $split\text{-}sound2: \langle splittable \ C \ Cs \Longrightarrow \forall \ C' \in fset \ Cs. \{C\} \models_{AFS} \{C'\} \rangle$ **and**
 $split\text{-}simp: \langle splittable \ C \ Cs \Longrightarrow C \in SRed_F$
 $(\{AF.Pair \ (F\text{-of} \ bot) \ (ffUnion \ ((|\cdot|) \ neg \ |\cdot| \ A\text{-of} \ |\cdot| \ Cs) \ |\cup| \ A\text{-of} \ C)\} \cup fset$
 $Cs) \rangle$
begin

Rule definitions follow a similar naming convention to our two inference rules BASE and UNSAT defined in *annotated-calculus*: X_simp is the definition of the simplification rule, while X_pre is some precondition which must hold for the rule to be applicable.

abbreviation $split\text{-}pre :: \langle ('f, 'v) \text{ AF} \Rightarrow ('f, 'v) \text{ AF fset} \Rightarrow \text{bool} \rangle$ **where**
 $\langle split\text{-}pre \ C \ Cs \equiv splittable \ C \ Cs \rangle$

abbreviation $split\text{-}res :: \langle ('f, 'v) \text{ AF} \Rightarrow ('f, 'v) \text{ AF fset} \Rightarrow ('f, 'v) \text{ AF set} \rangle$ **where**
 $\langle split\text{-}res \ C \ Cs \equiv$
 $(insert \ (AF.Pair \ (F\text{-of} \ bot) \ (ffUnion \ (fimage \ neg \ |\cdot| \ A\text{-of} \ |\cdot| \ Cs) \ |\cup| \ A\text{-of} \ C))$
 $(fset \ Cs)) \rangle$

abbreviation $split\text{-}simp :: \langle ('f, 'v) \text{ AF} \Rightarrow ('f, 'v) \text{ AF fset} \Rightarrow ('f, 'v) \text{ AF simplification} \rangle$ **where**
 $\langle split\text{-}simp \ C \ Cs \equiv Simplify \ \{C\} \ (split\text{-}res \ C \ Cs) \rangle$

inductive-set $Simps\text{-}with\text{-}Split :: \langle ('f, 'v) \text{ AF simplification set} \rangle$ **where**
 $split: \langle split\text{-}pre \ C \ Cs \Longrightarrow split\text{-}simp \ C \ Cs \in Simps\text{-}with\text{-}Split \rangle$
 $| \text{ other}: \langle simp \in Simps \Longrightarrow simp \in Simps\text{-}with\text{-}Split \rangle$

theorem $Inf\text{-}with\text{-}split\text{-}sound\text{-}wrt\text{-}entails\text{-}sound:$
 $\langle \iota \in Simps\text{-}with\text{-}Split \Longrightarrow \forall \ C \in S\text{-}to \ \iota. S\text{-}from \ \iota \models_{AFS} \{C\} \rangle$
 $\langle proof \rangle$

lemma $split\text{-}redundant: \langle split\text{-}pre \ C \ Cs \Longrightarrow C \in SRed_F \ (split\text{-}res \ C \ Cs) \rangle$
 $\langle proof \rangle$

lemma $simps\text{-}with\text{-}split\text{-}are\text{-}simps: \langle \iota \in Simps\text{-}with\text{-}Split \Longrightarrow (S\text{-}from \ \iota - S\text{-}to \ \iota) \rangle$

$\subseteq SRed_F (S\text{-to } \iota)$
 $\langle proof \rangle$

sublocale *AF-calc-ext*: *AF-calculus-extended bot SInf entails entails-sound*
 $SRed_I SRed_F Simps\text{-with-Split } OptInfs$
 $\langle proof \rangle$

end

locale *splitting-calculus* =
core: annotated-calculus bot Inf entails entails-sound Red_I Red_F fml asn
for *bot* :: *'f* **and**
Inf :: $\langle 'f \text{ inference set} \rangle$ **and**
entails :: $\langle 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models \rangle$ 50) **and**
entails-sound :: $\langle 'f \text{ set} \Rightarrow 'f \text{ set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_s \rangle$ 50) **and**
Red_I :: $\langle 'f \text{ set} \Rightarrow 'f \text{ inference set} \rangle$ **and**
Red_F :: $\langle 'f \text{ set} \Rightarrow 'f \text{ set} \rangle$ **and**
fml :: $\langle 'v :: \text{countable} \Rightarrow 'f \rangle$ **and**
asn :: $\langle 'f \text{ sign} \Rightarrow ('v :: \text{countable}) \text{ sign set} \rangle$
begin

interpretation *AF-sound-cons-rel*: *consequence-relation $\langle \text{to-}AF \text{ bot} \rangle \langle \models_{sAF} \rangle$*
 $\langle proof \rangle$

interpretation *SInf-sound-inf-system*: *sound-inference-system core.SInf $\langle \text{to-}AF \text{ bot} \rangle \langle \models_{sAF} \rangle$*
 $\langle proof \rangle$

Rule definitions follow a similar naming convention to our two inference rules BASE and UNSAT defined in *annotated-calculus*: *X_simp* is the definition of the simplification rule, while *X_pre* is some precondition which must hold for the rule to be applicable.

definition *split-form* :: $\langle 'f \Rightarrow 'f \text{ fset} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{split-form } C \text{ } Cs \longleftrightarrow C \neq \text{bot} \wedge \text{fcard } Cs \geq 2$
 $\wedge \{C\} \models_s \text{fset } Cs \wedge (\forall C'. C' \in Cs \longrightarrow C \in Red_F \{C'\}) \rangle$

definition *mk-split* :: $\langle 'f \Rightarrow 'f \text{ fset} \Rightarrow ('f, 'v) AF \text{ fset} \rangle$ **where**
 $\langle \text{split-form } C \text{ } Cs \Longrightarrow \text{mk-split } C \text{ } Cs \equiv (\lambda C'. AF.Pair C' \{ \mid \text{SOME } a. a \in \text{asn} (Pos C') \} \} \mid \mid Cs) \rangle$

definition *splittable* :: $\langle ('f, 'v) AF \Rightarrow ('f, 'v) AF \text{ fset} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{splittable } C \text{ } Cs \equiv \text{split-form } (F\text{-of } C) (F\text{-of } \mid \mid Cs) \wedge \text{mk-split } (F\text{-of } C) (F\text{-of } \mid \mid Cs) = Cs \rangle$

lemma *split-creates-singleton-assertion-sets*:
 $\langle \text{splittable } C \text{ } Cs \Longrightarrow A \in Cs \Longrightarrow (\exists a. A\text{-of } A = \{ \mid a \} \} \rangle$
 $\langle proof \rangle$

lemma *split-all-assertion-sets-asn*:

$\langle \text{splittable } C \ Cs \implies A \mid C \implies (\exists a. A\text{-of } A = \{|a|\} \wedge a \in \text{asn } (\text{Pos } (F\text{-of } A))) \rangle$
 $\langle \text{proof} \rangle$

lemma *split-all-pairs-in-As-in-Cs*: $\langle \text{splittable } C \ Cs \implies (\forall P. P \mid C \longrightarrow F\text{-of } P \mid (F\text{-of } |^{\dagger} C)) \rangle$
 $\langle \text{proof} \rangle$

lemma *split-all-pairs-in-Cs-in-As*:
 $\langle \text{splittable } C \ Cs \implies (\forall C. C \mid (F\text{-of } |^{\dagger} C) \longrightarrow (\exists a. AF.\text{Pair } C \{|a|\} \mid C)) \rangle$
 $\langle \text{proof} \rangle$

lemma *split-not-empty*: $\langle \text{splittable } C \ Cs \implies Cs \neq \{|\}\rangle$
 $\langle \text{proof} \rangle$

notation *core.sound-cons.entails-neg* (**infix** $\langle \models_{s_{\sim}} \rangle$ 50)

lemma *split-sound1*: $\langle \text{splittable } C \ Cs \implies \{C\} \models_{s_{AF}} \{AF.\text{Pair } bot \ (\text{ffUnion } (\text{fimage } neg \mid |^{\dagger} A\text{-of } |^{\dagger} Cs) \mid \cup \mid A\text{-of } C)\} \rangle$
 $\langle \text{proof} \rangle$

lemma *split-sound2*: $\langle \text{splittable } C \ Cs \implies \forall C' \in fset \ Cs. \{C\} \models_{s_{AF}} \{C'\} \rangle$
 $\langle \text{proof} \rangle$

lemma *split-simp*: $\langle \text{splittable } C \ Cs \implies C \in \text{core.SRed}_F \ (\{ AF.\text{Pair } bot \ (\text{ffUnion } ((|^{\dagger}) \text{ neg } \mid |^{\dagger} A\text{-of } |^{\dagger} Cs) \mid \cup \mid A\text{-of } C) \} \cup fset \ Cs) \rangle$
 $\langle \text{proof} \rangle$

sublocale *empty-simps*: *AF-calculus-extended to- AF bot* *core.SInf* $(\models_{AF})$
 $(\models_{s_{AF}})$ *core.SRed_I* *core.SRed_F* $\{\} \ \{\}$
 $\langle \text{proof} \rangle$

lemma *extend-simps-with-split*:

assumes

$\langle AF\text{-calculus-extended } (to\text{-}AF \text{ bot}) \text{ core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I$
 $\text{core.SRed}_F \text{ Simps OptInfs} \rangle$

shows

$\langle AF\text{-calculus-with-split } (to\text{-}AF \text{ bot}) \text{ core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I \text{ core.SRed}_F$
 Simps OptInfs
 $\text{splittable} \rangle$
 $\langle \text{proof} \rangle$

interpretation *splitting-calc-with-split*:

AF-calculus-with-split to- AF bot *core.SInf* $(\models_{AF})$ $(\models_{s_{AF}})$ *core.SRed_I* *core.SRed_F*
 $\{\} \ \{\}$ *splittable*
 $\langle \text{proof} \rangle$

end

9.1.2 The Collect Rule

locale *AF-calculus-with-collect* = *base-calculus: AF-calculus-extended* *bot SInf*
entails entails-sound SRed_I SRed_F Simps OptInfs
for *bot* :: $\langle ('f, 'v :: \text{countable}) \text{ AF} \rangle$ **and**
SInf :: $\langle ('f, 'v) \text{ AF inference set} \rangle$ **and**
entails :: $\langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{AF} \rangle$ 50) **and**
entails-sound :: $\langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{sAF} \rangle$ 50)
and
SRed_I :: $\langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF inference set} \rangle$ **and**
SRed_F :: $\langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \rangle$ **and**
Simps :: $\langle ('f, 'v) \text{ AF simplification set} \rangle$ **and**
OptInfs :: $\langle ('f, 'v) \text{ AF inference set} \rangle$
+ **assumes**
collect-redundant: $\langle F\text{-of } C \neq F\text{-of } bot \wedge \mathcal{M} \models_{AF} \{AF.Pair (F\text{-of } bot) (A\text{-of } C)\} \wedge$
 $(\forall C \in \mathcal{M}. F\text{-of } C = F\text{-of } bot) \implies C \in SRed_F \mathcal{M} \rangle$
begin

interpretation *AF-sound-cons-rel*: *consequence-relation bot* $\langle \models_{sAF} \rangle$
 $\langle \text{proof} \rangle$

interpretation *SInf-sound-inf-system*: *sound-inference-system SInf bot* $\langle \models_{sAF} \rangle$
 $\langle \text{proof} \rangle$

abbreviation *collect-pre* :: $\langle ('f, 'v) \text{ AF} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{collect-pre } C \mathcal{M} \equiv$
 $F\text{-of } C \neq F\text{-of } bot \wedge \mathcal{M} \models_{AF} \{AF.Pair (F\text{-of } bot) (A\text{-of } C)\} \wedge (\forall C \in \mathcal{M}.$
 $F\text{-of } C = F\text{-of } bot) \rangle$

abbreviation *collect-simp* :: $\langle ('f, 'v) \text{ AF} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF simplification} \rangle$ **where**
 $\langle \text{collect-simp } C \mathcal{M} \equiv \text{Simplify } (\text{insert } C \mathcal{M}) \mathcal{M} \rangle$

inductive-set *Simps-with-Collect* :: $\langle ('f, 'v) \text{ AF simplification set} \rangle$ **where**
collect: $\langle \text{collect-pre } C \mathcal{M} \implies \text{collect-simp } C \mathcal{M} \in \text{Simps-with-Collect} \rangle$
other: $\langle \iota \in \text{Simps} \implies \iota \in \text{Simps-with-Collect} \rangle$

theorem *SInf-with-collect-sound-wrt-entails-sound*:
 $\langle \iota \in \text{Simps-with-Collect} \implies \forall C \in S\text{-to } \iota. S\text{-from } \iota \models_{sAF} \{C\} \rangle$
 $\langle \text{proof} \rangle$

theorem *simps-with-collect-are-simps*:

$\langle \iota \in \text{Simps-with-Collect} \implies (S\text{-from } \iota - S\text{-to } \iota) \subseteq SRed_F (S\text{-to } \iota) \rangle$
 $\langle \text{proof} \rangle$

sublocale *AF-calc-ext*: *AF-calculus-extended bot SInf entails entails-sound SRed_I*
SRed_F

Simps-with-Collect OptInfs
 $\langle \text{proof} \rangle$

end

context *splitting-calculus*
begin

lemma *collect-redundant*: $\langle F\text{-of } \mathcal{C} \neq \text{bot} \wedge \mathcal{M} \models_{AF} \{AF.Pair \text{ bot } (A\text{-of } \mathcal{C})\} \wedge$
 $(\forall C \in \mathcal{M}. F\text{-of } C = \text{bot}) \implies C \in \text{core.SRed}_F \mathcal{M} \rangle$
 $\langle \text{proof} \rangle$

lemma *extend-simps-with-collect*:

assumes

$\langle AF\text{-calculus-extended } (to\text{-AF bot}) \text{ core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I$
 $\text{core.SRed}_F \text{ Simps OptInfs} \rangle$

shows

$\langle AF\text{-calculus-with-collect } (to\text{-AF bot}) \text{ core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I$
 $\text{core.SRed}_F \text{ Simps OptInfs} \rangle$
 $\langle \text{proof} \rangle$

interpretation *splitting-calc-with-collect*:

AF-calculus-with-collect to-AF bot core.SInf $(\models_{AF}) (\models_{s_{AF}})$ *core.SRed_I core.SRed_F*
 $\{\} \{\}$
 $\langle \text{proof} \rangle$

end

9.1.3 The Trim Rule

locale *AF-calculus-with-trim* = *base-calculus*: *AF-calculus-extended bot SInf*
entails entails-sound SRed_I SRed_F Simps OptInfs

for *bot* :: $\langle ('f, 'v :: \text{countable}) AF \rangle$ **and**

SInf :: $\langle ('f, 'v) AF \text{ inference set} \rangle$ **and**

entails :: $\langle ('f, 'v) AF \text{ set} \Rightarrow ('f, 'v) AF \text{ set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{AF} \rangle$ 50) **and**

entails-sound :: $\langle ('f, 'v) AF \text{ set} \Rightarrow ('f, 'v) AF \text{ set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{s_{AF}} \rangle$ 50)

and

SRed_I :: $\langle ('f, 'v) AF \text{ set} \Rightarrow ('f, 'v) AF \text{ inference set} \rangle$ **and**

SRed_F :: $\langle ('f, 'v) AF \text{ set} \Rightarrow ('f, 'v) AF \text{ set} \rangle$ **and**

Simps :: $\langle ('f, 'v) AF \text{ simplification set} \rangle$ **and**

OptInfs :: $\langle ('f, 'v) AF \text{ inference set} \rangle$

+ **assumes**

trim-sound: $\langle A\text{-of } \mathcal{C} = A \mid \cup \mid B \wedge F\text{-of } \mathcal{C} \neq F\text{-of } \text{bot} \wedge$
 $\mathcal{M} \cup \{ AF.Pair (F\text{-of } \text{bot}) A \} \models_{s_{AF}} \{ AF.Pair (F\text{-of } \text{bot}) B \} \wedge$
 $(\forall \mathcal{C} \in \mathcal{M}. F\text{-of } \mathcal{C} = (F\text{-of } \text{bot})) \wedge A \mid \cap \mid B = \{\mid\} \wedge A \neq \{\mid\}$
 $\implies \text{insert } \mathcal{C} \mathcal{M} \models_{s_{AF}} \{ AF.Pair (F\text{-of } \mathcal{C}) B \} \rangle$ **and**
trim-redundant: $\langle A\text{-of } \mathcal{C} = A \mid \cup \mid B \wedge F\text{-of } \mathcal{C} \neq F\text{-of } \text{bot} \wedge$
 $\mathcal{M} \cup \{ AF.Pair (F\text{-of } \text{bot}) A \} \models_{s_{AF}} \{ AF.Pair (F\text{-of } \text{bot}) B \} \wedge$
 $(\forall \mathcal{C} \in \mathcal{M}. F\text{-of } \mathcal{C} = (F\text{-of } \text{bot})) \wedge A \mid \cap \mid B = \{\mid\} \wedge A \neq \{\mid\}$
 $\implies \mathcal{C} \in SRed_F (\mathcal{M} \cup \{ AF.Pair (F\text{-of } \mathcal{C}) B \}) \rangle$

begin

interpretation *AF-sound-cons-rel*: consequence-relation *bot* $\langle (\models_{s_{AF}}) \rangle$
 $\langle \text{proof} \rangle$

interpretation *SInf-sound-inf-system*: sound-inference-system *SInf* *bot* $\langle (\models_{s_{AF}}) \rangle$
 $\langle \text{proof} \rangle$

abbreviation *trim-pre* :: $\langle ('f, 'v) AF \Rightarrow 'v \text{ sign fset} \Rightarrow 'v \text{ sign fset} \Rightarrow ('f, 'v) AF$
 $\text{set} \Rightarrow \text{bool} \rangle$ **where**

$\langle \text{trim-pre } \mathcal{C} A B \mathcal{M} \equiv A\text{-of } \mathcal{C} = A \mid \cup \mid B \wedge F\text{-of } \mathcal{C} \neq F\text{-of } \text{bot} \wedge$
 $\mathcal{M} \cup \{ AF.Pair (F\text{-of } \text{bot}) A \} \models_{s_{AF}} \{ AF.Pair (F\text{-of } \text{bot}) B \}$
 $\} \wedge$
 $(\forall \mathcal{C} \in \mathcal{M}. F\text{-of } \mathcal{C} = (F\text{-of } \text{bot})) \wedge A \mid \cap \mid B = \{\mid\} \wedge A \neq \{\mid\} \rangle$

abbreviation *trim-simp* :: $\langle ('f, 'v) AF \Rightarrow 'v \text{ sign fset} \Rightarrow 'v \text{ sign fset} \Rightarrow ('f, 'v)$
 $AF \text{ set} \Rightarrow$

$(('f, 'v) AF \text{ simplification}) \rangle$ **where**
 $\langle \text{trim-simp } \mathcal{C} A B \mathcal{M} \equiv \text{Simplify } (\text{insert } \mathcal{C} \mathcal{M}) (\text{insert } (AF.Pair (F\text{-of } \mathcal{C}) B)$
 $\mathcal{M}) \rangle$

inductive-set *Simps-with-Trim* :: $\langle ('f, 'v) AF \text{ simplification set} \rangle$ **where**

trim: $\langle \text{trim-pre } \mathcal{C} A B \mathcal{M} \implies \text{trim-simp } \mathcal{C} A B \mathcal{M} \in \text{Simps-with-Trim} \rangle$
other: $\langle \iota \in \text{Simps} \implies \iota \in \text{Simps-with-Trim} \rangle$

theorem *SInf-with-trim-sound-wrt-entails-sound*:

$\langle \iota \in \text{Simps-with-Trim} \implies \forall \mathcal{C} \in S\text{-to } \iota. S\text{-from } \iota \models_{s_{AF}} \{\mathcal{C}\} \rangle$
 $\langle \text{proof} \rangle$

theorem *simps-with-trim-are-simps*:

$\langle \iota \in \text{Simps-with-Trim} \implies (S\text{-from } \iota - S\text{-to } \iota) \subseteq SRed_F (S\text{-to } \iota) \rangle$
 $\langle \text{proof} \rangle$

sublocale *AF-calc-ext*: *AF-calculus-extended* *bot* *SInf* entails entails-sound *SRed_I*
SRed_F

Simps-with-Trim OptInfs
 $\langle \text{proof} \rangle$

end

context *splitting-calculus*
begin

lemma *trim-sound*: $\langle A\text{-of } C' = A \mid \cup \mid B \wedge F\text{-of } C' \neq \text{bot} \wedge$
 $\mathcal{N} \cup \{AF.Pair \text{ bot } A\} \models_{s_{AF}} \{AF.Pair \text{ bot } B\} \wedge$
 $(\forall C \in \mathcal{N}. F\text{-of } C = \text{bot}) \wedge A \mid \cap \mid B = \{\mid\} \wedge A \neq \{\mid\}$
 $\implies C = AF.Pair (F\text{-of } C') B \implies \text{insert } C' \mathcal{N} \models_{s_{AF}} \{C\} \rangle$
 $\langle \text{proof} \rangle$

theorem *trim-redundant*: $\langle A\text{-of } C = A \mid \cup \mid B \wedge$
 $F\text{-of } C \neq \text{bot} \wedge$
 $\mathcal{M} \cup \{AF.Pair \text{ bot } A\} \models_{s_{AF}} \{AF.Pair \text{ bot } B\} \wedge$
 $(\forall C \in \mathcal{M}. F\text{-of } C = \text{bot}) \wedge A \mid \cap \mid B = \{\mid\} \wedge A \neq \{\mid\} \implies$
 $C \in \text{core.SRed}_F (\mathcal{M} \cup \{ AF.Pair (F\text{-of } C) B \}) \rangle$
 $\langle \text{proof} \rangle$

lemma *extend-simps-with-trim*:
assumes
 $\langle AF\text{-calculus-extended } (to\text{-}AF \text{ bot}) \text{ core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I$
 $\text{core.SRed}_F \text{ Simps OptInfs} \rangle$
shows
 $\langle AF\text{-calculus-with-trim } (to\text{-}AF \text{ bot}) \text{ core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I \text{ core.SRed}_F$
 $\text{Simps OptInfs} \rangle$
 $\langle \text{proof} \rangle$

interpretation *splitting-calc-with-trim*:
 $AF\text{-calculus-with-trim } to\text{-}AF \text{ bot } \text{core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I \text{ core.SRed}_F$
 $\{\} \{\}$
 $\langle \text{proof} \rangle$

end

9.2 Extra Inferences

We extend our basic splitting calculus with new optional rules:

- **STRONGUNSAT** is a variant of **UNSAT** which uses the sound entailment instead of the "normal" entailment;
- **APPROX** is a very special case for **SPLIT** where $n = 1$;
- **TAUTO** inserts a new formula which is always true.

9.2.1 The StrongUnsat Rule

locale *AF-calculus-with-strong-unsat* =
 base: *AF-calculus-extended bot SInf entails entails-sound Red-I Red-F Simps*
OptInfs
for *bot* :: $\langle ('f, 'v :: \text{countable}) \text{ AF} \rangle$ **and**
SInf :: $\langle ('f, 'v) \text{ AF inference set} \rangle$ **and**
entails :: $\langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow \text{bool} \rangle$ **and**
entails-sound :: $\langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{s_{AF}} \rangle$ 50)
and
Red-I :: $\langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF inference set} \rangle$ **and**
Red-F :: $\langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \rangle$ **and**
Simps :: $\langle ('f, 'v) \text{ AF simplification set} \rangle$ **and**
OptInfs :: $\langle ('f, 'v) \text{ AF inference set} \rangle$
begin

We follow the same naming conventions for our new inference rules as for the two inference rules defined in *annotated-calculus*. *X_inf* defines the inference rule, while *X_pre* is the precondition for the application of the inference rule.

abbreviation *strong-unsat-pre* :: $\langle ('f, 'v) \text{ AF list} \Rightarrow \text{bool} \rangle$ **where**
 $\langle \text{strong-unsat-pre } \mathcal{M} \equiv (\text{set } \mathcal{M} \models_{s_{AF}} \{\text{bot}\}) \wedge (\forall x \in \text{set } \mathcal{M}. F\text{-of } x = (F\text{-of } \text{bot})) \rangle$

abbreviation *strong-unsat-inf* :: $\langle ('f, 'v) \text{ AF list} \Rightarrow ('f, 'v) \text{ AF inference} \rangle$ **where**
 $\langle \text{strong-unsat-inf } \mathcal{M} \equiv \text{Infer } \mathcal{M} \text{ bot} \rangle$

Instead of considering an inference system with only the new rule, here STRONGUNSAT, we instead add it to the inference system provided so that it is possible to extend the system rule by rule in a modular way.

inductive-set *OptInfs-with-strong-unsat* :: $\langle ('f, 'v) \text{ AF inference set} \rangle$ **where**
strong-unsat: $\langle \text{strong-unsat-pre } \mathcal{M} \Longrightarrow \text{strong-unsat-inf } \mathcal{M} \in \text{OptInfs-with-strong-unsat} \rangle$
from-OptInf: $\langle \iota \in \text{OptInfs} \Longrightarrow \iota \in \text{OptInfs-with-strong-unsat} \rangle$

theorem *OptInf-with-strong-unsat-sound-wrt-entails-sound*:
 $\langle \iota \in \text{OptInfs-with-strong-unsat} \Longrightarrow \text{set } (\text{prems-of } \iota) \models_{s_{AF}} \{\text{concl-of } \iota\} \rangle$
 $\langle \text{proof} \rangle$

interpretation *AF-sound-cons-rel*: *consequence-relation bot* $\langle \models_{s_{AF}} \rangle$
 $\langle \text{proof} \rangle$

interpretation *SInf-sound-inf-system*: *sound-inference-system SInf bot* $\langle \models_{s_{AF}} \rangle$
 $\langle \text{proof} \rangle$

definition *Red-I-ext* **where**
 $\langle \text{Red-I-ext } \mathcal{N} = \text{Red-I } \mathcal{N} \cup \{ \text{strong-unsat-inf } \mathcal{M} \mid \mathcal{M}. \text{strong-unsat-pre } \mathcal{M} \wedge \text{bot} \in \mathcal{N} \} \rangle$

interpretation *AF-calc-with-strong-unsat:*

AF-calculus bot SInf entails entails-sound Red-I Red-F

⟨proof⟩

sublocale *AF-calc-ext: AF-calculus-extended bot SInf entails entails-sound Red-I Red-F*

Simps OptInfs-with-strong-unsat

⟨proof⟩

end

context *splitting-calculus*

begin

lemma *extend-infs-with-strong-unsat:*

assumes

⟨AF-calculus-extended (to-AF bot) core.SInf ($\models_{AF}$) ($\models_{s_{AF}}$) core.SRed_I core.SRed_F Simps OptInfs⟩

shows

⟨AF-calculus-with-strong-unsat (to-AF bot) core.SInf ($\models_{AF}$) ($\models_{s_{AF}}$) core.SRed_I core.SRed_F Simps

OptInfs⟩

⟨proof⟩

interpretation *splitting-calc-with-strong-unsat: AF-calculus-with-strong-unsat to-AF bot core.SInf*

($\models_{AF}$) ($\models_{s_{AF}}$) core.SRed_I core.SRed_F {} {}

⟨proof⟩

end

9.2.2 The Tauto Rule

locale *AF-calculus-with-tauto =*

base: AF-calculus-extended bot SInf entails entails-sound Red-I Red-F Simps OptInfs

for *bot :: ⟨('f, 'v :: countable) AF⟩ and*

SInf :: ⟨('f, 'v) AF inference set⟩ and

entails :: ⟨('f, 'v) AF set $\Rightarrow$ ('f, 'v) AF set $\Rightarrow$ bool⟩ and

entails-sound :: ⟨('f, 'v) AF set $\Rightarrow$ ('f, 'v) AF set $\Rightarrow$ bool⟩ (infix $\langle \models_{s_{AF}} \rangle$ 50)

and

Red-I :: ⟨('f, 'v) AF set $\Rightarrow$ ('f, 'v) AF inference set⟩ and

Red-F :: ⟨('f, 'v) AF set $\Rightarrow$ ('f, 'v) AF set⟩ and

Simps :: ⟨('f, 'v) AF simplification set⟩ and

OptInfs :: ⟨('f, 'v) AF inference set⟩

begin

abbreviation *tauto-pre :: ⟨('f, 'v) AF $\Rightarrow$ bool⟩ where*

⟨tauto-pre C $\equiv$ {} $\models_{s_{AF}}$ { C }⟩

abbreviation *tauto-inf* :: $\langle ('f, 'v) AF \Rightarrow ('f, 'v) AF \text{ inference} \rangle$ **where**
 $\langle \text{tauto-inf } C \equiv \text{Infer } [] C \rangle$

inductive-set *OptInfs-with-tauto* :: $\langle ('f, 'v) AF \text{ inference set} \rangle$ **where**
 $\text{tauto: } \langle \text{tauto-pre } C \Longrightarrow \text{tauto-inf } C \in \text{OptInfs-with-tauto} \rangle$
 $| \text{from-OptInfs: } \langle \iota \in \text{OptInfs} \Longrightarrow \iota \in \text{OptInfs-with-tauto} \rangle$

theorem *OptInfs-with-tauto-sound-wrt-entails-sound*: $\langle \iota \in \text{OptInfs-with-tauto} \Longrightarrow$

$\text{set } (\text{prems-of } \iota) \models_{sAF} \{ \text{concl-of } \iota \} \rangle$
 $\langle \text{proof} \rangle$

sublocale *AF-calc-ext*: *AF-calculus-extended bot SInf entails* $(\models_{sAF})$ *Red-I*
Red-F Simps OptInfs-with-tauto
 $\langle \text{proof} \rangle$

end

context *splitting-calculus*
begin

lemma *extend-infs-with-tauto*:

assumes

$\langle \text{AF-calculus-extended } (\text{to-}AF \text{ bot}) \text{ core.SInf } (\models_{AF}) (\models_{sAF}) \text{ core.SRed}_I$
 $\text{core.SRed}_F \text{ Simps OptInfs} \rangle$

shows

$\langle \text{AF-calculus-with-tauto } (\text{to-}AF \text{ bot}) \text{ core.SInf } (\models_{AF}) (\models_{sAF}) \text{ core.SRed}_I \text{ core.SRed}_F$
 $\text{Simps OptInfs} \rangle$
 $\langle \text{proof} \rangle$

interpretation *splitting-calc-with-tauto*:

AF-calculus-with-tauto to-}AF bot core.SInf $(\models_{AF}) (\models_{sAF})$ *core.SRed}_I core.SRed}_F*
 $\{ \} \{ \}$
 $\langle \text{proof} \rangle$

end

9.2.3 The Approx Rule

locale *AF-calculus-with-approx* =

base: AF-calculus-extended bot SInf entails entails-sound Red-I Red-F Simps
OptInfs

for *bot* :: $\langle ('f, 'v :: \text{countable}) AF \rangle$ **and**

SInf :: $\langle ('f, 'v) AF \text{ inference set} \rangle$ **and**

$entails :: \langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow \text{bool} \rangle$ **and**
 $entails\text{-sound} :: \langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{s_{AF}} \rangle$ 50)

and

$Red\text{-}I :: \langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF inference set} \rangle$ **and**
 $Red\text{-}F :: \langle ('f, 'v) \text{ AF set} \Rightarrow ('f, 'v) \text{ AF set} \rangle$ **and**
 $Simps :: \langle ('f, 'v) \text{ AF simplification set} \rangle$ **and**
 $OptInfs :: \langle ('f, 'v) \text{ AF inference set} \rangle$

+ fixes

$approximates :: \langle 'v \text{ sign} \Rightarrow ('f, 'v) \text{ AF} \Rightarrow \text{bool} \rangle$
assumes
 $approx\text{-sound}: \langle approximates \ a \ C \Longrightarrow \{C\} \models_{s_{AF}} \{AF.Pair \ (F\text{-of} \ bot) \ (finsert \ (neg \ a) \ (A\text{-of} \ C))\} \rangle$

begin

abbreviation $approx\text{-pre} :: \langle 'v \text{ sign} \Rightarrow ('f, 'v) \text{ AF} \Rightarrow \text{bool} \rangle$ **where**
 $\langle approx\text{-pre} \ a \ C \equiv approximates \ a \ C \rangle$

abbreviation $approx\text{-inf} :: \langle ('f, 'v) \text{ AF} \Rightarrow 'v \text{ sign} \Rightarrow ('f, 'v) \text{ AF inference} \rangle$ **where**
 $\langle approx\text{-inf} \ C \ a \equiv Infer \ [C] \ (AF.Pair \ (F\text{-of} \ bot) \ (finsert \ (neg \ a) \ (A\text{-of} \ C))) \rangle$

inductive-set $OptInfs\text{-with-approx} :: \langle ('f, 'v) \text{ AF inference set} \rangle$ **where**
 $approx: \langle approx\text{-pre} \ a \ C \Longrightarrow approx\text{-inf} \ C \ a \in OptInfs\text{-with-approx} \rangle$
 $| \text{from-OptInfs}: \langle \iota \in OptInfs \Longrightarrow \iota \in OptInfs\text{-with-approx} \rangle$

theorem $OptInfs\text{-with-approx-sound-wrt-entails-sound}$:
 $\langle \iota \in OptInfs\text{-with-approx} \Longrightarrow set \ (prems\text{-of} \ \iota) \models_{s_{AF}} \{concl\text{-of} \ \iota\} \rangle$
 $\langle proof \rangle$

sublocale $AF\text{-calc-ext}$: $AF\text{-calculus-extended} \ bot \ SInf \ entails \ (\models_{s_{AF}}) \ Red\text{-}I \ Red\text{-}F \ Simps$
 $OptInfs\text{-with-approx}$
 $\langle proof \rangle$

end

context $splitting\text{-calculus}$
begin

definition $approximates :: \langle 'v \text{ sign} \Rightarrow ('f, 'v) \text{ AF} \Rightarrow \text{bool} \rangle$ **where**
 $\langle approximates \ a \ C \equiv a \in asn \ (Pos \ (F\text{-of} \ C)) \rangle$

lemma $approx\text{-sound}$: $\langle approximates \ a \ C \Longrightarrow \{C\} \models_{s_{AF}} \{AF.Pair \ bot \ (finsert \ (neg \ a) \ (A\text{-of} \ C))\} \rangle$
 $\langle proof \rangle$

lemma $extend\text{-infs-with-approx}$:
assumes

$\langle AF\text{-calculus-extended } (to\text{-}AF \text{ bot}) \text{ core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I \text{ core.SRed}_F$
 Simps
 $\text{OptInfs} \rangle$
shows
 $\langle AF\text{-calculus-with-approx } (to\text{-}AF \text{ bot}) \text{ core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I$
 $\text{core.SRed}_F \text{ Simps}$
 $\text{OptInfs approximates} \rangle$
 $\langle \text{proof} \rangle$

interpretation *splitting-calc-with-approx:*
 $AF\text{-calculus-with-approx } to\text{-}AF \text{ bot } \text{core.SInf } (\models_{AF}) (\models_{s_{AF}}) \text{ core.SRed}_I \text{ core.SRed}_F$
 $\{\} \{\}$
 approximates
 $\langle \text{proof} \rangle$

end

9.3 Combining all simplifications and optional inferences

We have augmented the core calculus with each simplification and optional rule separately. We now show how to augment the core calculus with all of them in succession.

context *splitting-calculus*
begin

interpretation *with-A: AF-calculus-with-approx to-AF bot* $\text{core.SInf } (\models_{AF}) (\models_{s_{AF}})$
 core.SRed_I
 $\text{core.SRed}_F \{\} \{\} \text{ approximates}$
 $\langle \text{proof} \rangle$

interpretation *with-AT: AF-calculus-with-tauto to-AF bot* $\text{core.SInf } (\models_{AF}) (\models_{s_{AF}})$
 core.SRed_I
 $\text{core.SRed}_F \{\} \text{ with-A.OptInfs-with-approx}$
 $\langle \text{proof} \rangle$

interpretation *with-ATS: AF-calculus-with-strong-unsat to-AF bot* $\text{core.SInf } (\models_{AF})$
 $(\models_{s_{AF}})$
 $\text{core.SRed}_I \text{ core.SRed}_F \{\} \text{ with-AT.OptInfs-with-tauto}$
 $\langle \text{proof} \rangle$

interpretation *with-ATS-T: AF-calculus-with-trim to-AF bot* $\text{core.SInf } (\models_{AF})$
 $(\models_{s_{AF}}) \text{ core.SRed}_I$
 $\text{core.SRed}_F \{\} \text{ with-ATS.OptInfs-with-strong-unsat}$
 $\langle \text{proof} \rangle$

interpretation *with-ATS-TC: AF-calculus-with-collect to-AF bot* $\text{core.SInf } (\models_{AF})$
 $(\models_{s_{AF}})$
 $\text{core.SRed}_I \text{ core.SRed}_F \text{ with-ATS-T.Simps-with-Trim with-ATS.OptInfs-with-strong-unsat}$
 $\langle \text{proof} \rangle$

interpretation *with-all*: *AF-calculus-with-split to-AF bot core.SInf* ($\models_{AF}$) ($\models_{s_{AF}}$)

core.SRed_I core.SRed_F with-ATS-TC.Simps-with-Collect with-ATS.OptInfs-with-strong-unsat
splittable
 ⟨proof⟩

interpretation *full-splitting-calculus*: *AF-calculus-extended to-AF bot*
core.SInf ($\models_{AF}$) ($\models_{s_{AF}}$) *core.SRed_I core.SRed_F with-all.Simps-with-Split*
with-ATS.OptInfs-with-strong-unsat
 ⟨proof⟩

Simplifications and optional inferences can be integrated in derivations.
 This is made obvious by the following two lemmas.

lemma *simp-in-derivations*: $\langle \delta \in \text{with-all.Simps-with-Split} \implies$
core.S-calculus.derive ($\mathcal{M} \cup S\text{-from } \delta$) ($\mathcal{M} \cup S\text{-to } \delta$)
 ⟨proof⟩

lemma *opt-infs-in-derivations*: $\langle \iota \in \text{with-ATS.OptInfs-with-strong-unsat} \implies$
core.S-calculus.derive ($\mathcal{M} \cup \text{set (prems-of } \iota)$) ($\mathcal{M} \cup \text{set (prems-of } \iota) \cup \{\text{concl-of } \iota\}$)
 ⟨proof⟩

end

9.4 The BinSplit Simplification Rule

For the sake of the Lightweight Avatar calculus, we define the BINSPLIT simplification rule, and show that it is indeed a sound simplification rule as in the case of the Split rule.

locale *AF-calculus-with-binsplit* =
base-calculus: *AF-calculus-extended bot SInf entails entails-sound SRed_I*
SRed_F Simps OptInfs
for *bot* :: $\langle ('f, 'v) :: \text{countable} \rangle AF \rangle$ **and**
SInf :: $\langle ('f, 'v) AF \text{ inference set} \rangle$ **and**
entails :: $\langle ('f, 'v) AF \text{ set} \Rightarrow ('f, 'v) AF \text{ set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{AF} \rangle$ 50) **and**
entails-sound :: $\langle ('f, 'v) AF \text{ set} \Rightarrow ('f, 'v) AF \text{ set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{s_{AF}} \rangle$ 50)
and
SRed_I :: $\langle ('f, 'v) AF \text{ set} \Rightarrow ('f, 'v) AF \text{ inference set} \rangle$ **and**
SRed_F :: $\langle ('f, 'v) AF \text{ set} \Rightarrow ('f, 'v) AF \text{ set} \rangle$ **and**
Simps :: $\langle ('f, 'v) AF \text{ simplification set} \rangle$ **and**
OptInfs :: $\langle ('f, 'v) AF \text{ inference set} \rangle$
+ fixes
bin-splittable :: $\langle ('f, 'v) AF \Rightarrow 'f \Rightarrow 'f \Rightarrow ('f, 'v) AF \text{ fset} \Rightarrow \text{bool} \rangle$
assumes
binsplit-prem-entails-cons: $\langle \text{bin-splittable } C \ C1 \ C2 \ Cs \implies \forall C' \in \text{fset } Cs. \{C\}$
 $\models_{s_{AF}} \{C'\} \rangle$ **and**
binsplit-simp: $\langle \text{bin-splittable } C \ C1 \ C2 \ Cs \implies C \in SRed_F (\text{fset } Cs) \rangle$
begin

We use the same naming convention as for the other rules, where X_pre is the condition which must hold for the rule to be applicable, X_res is its resultant and X_simp is the simplification rule itself.

abbreviation $bin_split_pre :: \langle ('f, 'v) AF \Rightarrow 'f \Rightarrow 'f \Rightarrow ('f, 'v) AF fset \Rightarrow bool \rangle$
where

$\langle bin_split_pre \ C \ C1 \ C2 \ Cs \equiv bin_splittable \ C \ C1 \ C2 \ Cs \rangle$

abbreviation $bin_split_res :: \langle ('f, 'v) AF \Rightarrow ('f, 'v) AF fset \Rightarrow ('f, 'v) AF set \rangle$
where

$\langle bin_split_res \ C \ Cs \equiv fset \ Cs \rangle$

abbreviation $bin_split_simp :: \langle ('f, 'v) AF \Rightarrow ('f, 'v) AF fset \Rightarrow ('f, 'v) AF simplification \rangle$ **where**

$\langle bin_split_simp \ C \ Cs \equiv Simplify \ \{C\} \ (bin_split_res \ C \ Cs) \rangle$

inductive-set $Simps_with_BinSplit :: \langle ('f, 'v) AF simplification set \rangle$ **where**

$binsplit: \langle bin_split_pre \ C \ C1 \ C2 \ Cs \implies bin_split_simp \ C \ Cs \in Simps_with_BinSplit \rangle$

| $other: \langle simp \in Simps \implies simp \in Simps_with_BinSplit \rangle$

theorem $SInf_with_binsplit_sound_wrt_entails_sound:$

$\langle \iota \in Simps_with_BinSplit \implies \forall \ C \in S_to \ \iota. \ S_from \ \iota \models_{sAF} \{C\} \rangle$
 $\langle proof \rangle$

lemma $binsplit_redundant: \langle bin_split_pre \ C \ C1 \ C2 \ Cs \implies C \in SRed_F \ (bin_split_res \ C \ Cs) \rangle$

$\langle proof \rangle$

lemma $simps_with_binsplit_are_simps:$

$\langle \iota \in Simps_with_BinSplit \implies (S_from \ \iota - S_to \ \iota) \subseteq SRed_F \ (S_to \ \iota) \rangle$
 $\langle proof \rangle$

sublocale $AF_calc_ext: AF_calculus_extended \ bot \ SInf \ entails \ entails_sound$

$SRed_I \ SRed_F \ Simps_with_BinSplit \ OptInfs$

$\langle proof \rangle$

end

context $splitting_calculus$

begin

definition $mk_bin_split :: \langle ('f, 'v) AF \Rightarrow 'f \Rightarrow 'f \Rightarrow ('f, 'v) AF fset \rangle$ **where**

$\langle split_form \ (F_of \ C) \ \{|C1, C2|\} \implies mk_bin_split \ C \ C1 \ C2 \equiv$

$let \ a = (SOME \ a. \ a \in asn \ (sign.Pos \ C1))$

in $\{|AF.Pair\ C1\ (finsert\ a\ (A-of\ C)),\ AF.Pair\ C2\ (finsert\ (neg\ a)\ (A-of\ C))|\}$

definition *bin-splittable* :: $\langle ('f, 'v)\ AF \Rightarrow 'f \Rightarrow 'f \Rightarrow ('f, 'v)\ AF\ fset \Rightarrow bool \rangle$
where
 $\langle bin-splittable\ C\ C1\ C2\ Cs \equiv split-form\ (F-of\ C)\ \{|C1,\ C2|\} \wedge mk-bin-split\ C\ C1\ C2 = Cs \rangle$

theorem *binsplit-prem-entails-cons*: $\langle bin-splittable\ C\ C1\ C2\ Cs \implies \forall\ C' \in fset\ Cs.\ \{C\} \models_{s_{AF}} \{C'\} \rangle$
 $\langle proof \rangle$

theorem *binsplit-simp*: $\langle bin-splittable\ C\ C1\ C2\ Cs \implies C \in core.SRed_F\ (fset\ Cs) \rangle$
 $\langle proof \rangle$

lemma *extend-simps-with-binsplit*:

assumes

$\langle AF-calculus-extended\ (to-AF\ bot)\ core.SInf\ (\models_{AF})\ (\models_{s_{AF}})\ core.SRed_I\ core.SRed_F\ Simps\ OptInfs \rangle$

shows

$\langle AF-calculus-with-binsplit\ (to-AF\ bot)\ core.SInf\ (\models_{AF})\ (\models_{s_{AF}})\ core.SRed_I\ core.SRed_F\ Simps\ OptInfs\ bin-splittable \rangle$
 $\langle proof \rangle$

interpretation *splitting-calc-with-binsplit*: *AF-calculus-with-binsplit to-AF bot core.SInf*
 $(\models_{AF})$
 $(\models_{s_{AF}})\ core.SRed_I\ core.SRed_F\ \{\}\ \{\}\ bin-splittable$
 $\langle proof \rangle$

end

end

theory *Lightweight-Avatar*

imports

Main

Modular-Splitting-Calculus

Saturation-Framework-Extensions.FO-Ordered-Resolution-Prover-Revisited

begin

9.5 Ordered Resolution with a Disjunctive Consequence Relation

locale *FO-resolution-prover-disjunctive* = *FO-resolution-prover S subst-atm id-subst comp-subst*
renaming-aparts atm-of-atms mgu less-atm

```

for
   $S :: \langle 'a :: \text{wellorder} \rangle \text{ clause} \Rightarrow 'a \text{ clause} \rangle$  and
   $\text{subst-atm} :: \langle 'a \Rightarrow 's \Rightarrow 'a \rangle$  and
   $\text{id-subst} :: \langle 's \rangle$  and
   $\text{comp-subst} :: \langle 's \Rightarrow 's \Rightarrow 's \rangle$  and
   $\text{renaming-aparts} :: \langle 'a \text{ clause list} \Rightarrow 's \text{ list} \rangle$  and
   $\text{atm-of-atms} :: \langle 'a \text{ list} \Rightarrow 'a \rangle$  and
   $\text{mgu} :: \langle 'a \text{ set set} \Rightarrow 's \text{ option} \rangle$  and
   $\text{less-atm} :: \langle 'a \Rightarrow 'a \Rightarrow \text{bool} \rangle$ 
begin

no-notation  $\text{entails-clss}$  (infix  $\langle \models_e \rangle$  50)
no-notation  $\text{Sema.entailment}$  ( $\langle (- \models / -) \rangle$  [53, 53] 53)
no-notation  $\text{Linear-Temporal-Logic-on-Streams.HLD-next}$  (infixr  $\cdot$  65)

notation  $\text{entails-clss}$  (infix  $\langle \models_{\cap e} \rangle$  50)

interpretation  $\text{gr}$ : ground-resolution-with-selection  $S\text{-}M\ S\ M$ 
   $\langle \text{proof} \rangle$ 

interpretation  $G$ : Soundness.sound-inference-system  $G\text{-}Inf\ M\ \{\{\#\}\}$  ( $\models_{\cap e}$ )
   $\langle \text{proof} \rangle$ 

interpretation  $G$ : clausal-counterex-reducing-inference-system  $G\text{-}Inf\ M\ \text{gr.INTERP}$ 
   $M$ 
   $\langle \text{proof} \rangle$ 

interpretation  $G$ : calculus-with-standard-redundancy  $\langle G\text{-}Inf\ M \rangle\ \langle \{\{\#\}\} \rangle\ \langle \models_{\cap e} \rangle$ 
   $\langle \langle < \rangle :: 'a \text{ clause} \Rightarrow 'a \text{ clause} \Rightarrow \text{bool} \rangle$ 
   $\langle \text{proof} \rangle$ 

interpretation  $G$ : clausal-counterex-reducing-calculus-with-standard-redundancy  $G\text{-}Inf$ 
   $M$ 
   $\text{gr.INTERP}\ M$ 
   $\langle \text{proof} \rangle$ 

interpretation  $G$ : Calculus.statically-complete-calculus  $\{\{\#\}\}\ G\text{-}Inf\ M\ (\models_{\cap e})$ 
   $G\text{-}Red\text{-}I\ M\ G\text{-}Red\text{-}F$ 
   $\langle \text{proof} \rangle$ 

sublocale  $F$ : lifting-intersection  $F\text{-}Inf\ \{\{\#\}\}\ UNIV\ G\text{-}Inf\ \lambda N. (\models_{\cap e})\ G\text{-}Red\text{-}I$ 
   $\lambda N. G\text{-}Red\text{-}F$ 
   $\{\{\#\}\}\ \lambda N. G\text{-}F\ G\text{-}I\text{-}opt\ \lambda D\ C\ C'. \text{False}$ 
   $\langle \text{proof} \rangle$ 

notation  $F.\text{entails-}\mathcal{G}$  (infix  $\models_{\cap \mathcal{G} e}$  50)

```

sublocale F : *sound-inference-system* $F\text{-Inf}$ $\{\{\#\}\}$ $(\models_{\cap \mathcal{G}e})$
 $\langle \text{proof} \rangle$

lemma $F\text{-stat-comp-calc}$: $\langle \text{Calculus.statically-complete-calculus} \{\{\#\}\} F\text{-Inf} (\models_{\cap \mathcal{G}e})$
 $F\text{-Red-I-}\mathcal{G}$
 $F\text{-Red-F-}\mathcal{G}\text{-empty} \rangle$
 $\langle \text{proof} \rangle$

sublocale F : $\text{Calculus.statically-complete-calculus} \{\{\#\}\} F\text{-Inf} (\models_{\cap \mathcal{G}e}) F\text{-Red-I-}\mathcal{G}$
 $F\text{-Red-F-}\mathcal{G}\text{-empty}$
 $\langle \text{proof} \rangle$

sublocale F' : $\text{Calculus.statically-complete-calculus} \{\{\#\}\} F\text{-Inf} (\models_{\cap \mathcal{G}e}) F\text{-empty-ord.Red-Red-I}$
 $F\text{-Red-F-}\mathcal{G}\text{-empty}$
 $\langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle \langle \text{proof} \rangle$

$(\models_{\cap \mathcal{G}e})$ is a conjunctive entailment, meaning that for $M \models_{\cap \mathcal{G}e} N$ to hold, each clause in N must be entailed by M . Unfortunately, this clashes with requirement (D3) $\llbracket \text{Disjunctive-Consequence-Relations.consequence-relation} \text{?bot ?entails; ?}M' \subseteq \text{?}M; \text{?}N' \subseteq \text{?}N; \text{?entails ?}M' \text{?}N \rrbracket \implies \text{?entails ?}M \text{?}N$ of a splitting calculus.

Therefore, we define a disjunctive version of this entailment by stating that $M \models_{\cup \mathcal{G}e} N$ iff there is some $C \in N$ such that $M \models_{\cap \mathcal{G}e} \{C\}$. This definition is not quite enough because it does not capture (D1) $\llbracket \text{Disjunctive-Consequence-Relations.consequence-relation} \text{?bot ?entails} \implies \text{?entails} \{ \text{?bot} \} \rrbracket$. More specifically, if N is empty, then there does not exist a $C \in N$! But we know that $M \models_{\cup \mathcal{G}e} \{\}$ if M is unsatisfiable. Hence $M \models_{\cup \mathcal{G}e} N$ if M is unsatisfiable, or there exists some $C \in N$ such that $M \models_{\cap \mathcal{G}e} \{C\}$. In addition, it is necessary to modify this definition to capture (D5) the compactness property of disjunctive entailment.

definition $\text{entails-}\mathcal{G}\text{-disj} :: \langle 'a \text{ clause set} \Rightarrow 'a \text{ clause set} \Rightarrow \text{bool} \rangle$ (**infix** $\langle \models_{\cup \mathcal{G}e} \rangle$ 50) **where**

$\langle M \models_{\cup \mathcal{G}e} N \longleftrightarrow M \models_{\cap \mathcal{G}e} \{\{\#\}\} \vee (\exists M' \subseteq M. \text{finite } M' \wedge (\exists C \in N. M' \models_{\cap \mathcal{G}e} \{C\})) \rangle$

$\langle \text{proof} \rangle$

lemma $\text{entails-}\mathcal{G}\text{-disj-subsets}$: $\langle M' \subseteq M \implies N' \subseteq N \implies M' \models_{\cup \mathcal{G}e} N' \implies M \models_{\cup \mathcal{G}e} N \rangle$
 $\langle \text{proof} \rangle$

lemma $\text{entails-}\mathcal{G}\text{-disj-compactness}$:

$\langle M \models_{\cup \mathcal{G}e} N \implies \exists M' N'. M' \subseteq M \wedge N' \subseteq N \wedge \text{finite } M' \wedge \text{finite } N' \wedge M' \models_{\cup \mathcal{G}e} N' \rangle$
 $\langle \text{proof} \rangle$

lemma *entails- $\mathcal{G}$ -disj-cut*: $\langle M \models_{\mathcal{G}e} N \cup \{C\} \implies M' \cup \{C\} \models_{\mathcal{G}e} N' \implies M \cup M' \models_{\mathcal{G}e} N \cup N' \rangle$
 $\langle proof \rangle$

lemma *entails- $\mathcal{G}$ -disj-cons-rel-ext*: $\langle consequence_relation \ \{\#\} \ (\models_{\mathcal{G}e}) \rangle$
 $\langle proof \rangle$

sublocale *entails- $\mathcal{G}$ -disj-cons-rel*: $consequence_relation \ \{\#\} \ (\models_{\mathcal{G}e})$
 $\langle proof \rangle$

notation *entails- $\mathcal{G}$ -disj-cons-rel.entails-neg* (**infix** $\langle \models_{\mathcal{G}e\sim} \ 50 \rangle$)

lemma *all-redundant-to-bottom*: $\langle C \neq \{\#\} \implies C \in F.Red-F-\mathcal{G}-empty \ \{\{\#\}\} \rangle$
 $\langle proof \rangle$

lemma *bottom-never-redundant*: $\langle \{\#\} \notin F.Red-F-\mathcal{G}-empty \ N \rangle$
 $\langle proof \rangle$

lemma $\langle F.Inf-between \ UNIV \ (F.Red-F-\mathcal{G}-empty \ N) \subseteq F.empty-ord.Red-Red-I \ N \rangle$
 $\langle proof \rangle$

end

9.6 Lightweight Avatar without BinSplit

Since the set $\mathbb{P}$ of nullary predicates is left unspecified, we cannot define *fml* nor *asn*. Therefore, we keep them abstract and leave it to anybody instantiating this locale to specify them.

locale *LA-calculus = FO-resolution-prover-disjunctive S subst-atm id-subst comp-subst renaming-aparts*

atm-of-atms mgu less-atm

for

S :: $\langle ('a :: wellorder) \ clause \Rightarrow 'a \ clause \rangle$ **and**

subst-atm :: $\langle 'a \Rightarrow 's \Rightarrow 'a \rangle$ **and**

id-subst :: $\langle 's \rangle$ **and**

comp-subst :: $\langle 's \Rightarrow 's \Rightarrow 's \rangle$ **and**

renaming-aparts :: $\langle 'a \ clause \ list \Rightarrow 's \ list \rangle$ **and**

atm-of-atms :: $\langle 'a \ list \Rightarrow 'a \rangle$ **and**

mgu :: $\langle 'a \ set \ set \Rightarrow 's \ option \rangle$ **and**

less-atm :: $\langle 'a \Rightarrow 'a \Rightarrow bool \rangle$

+ fixes

asn :: $\langle 'a \ clause \ sign \Rightarrow ('v :: countable) \ sign \ set \rangle$ **and**

fml :: $\langle 'v \Rightarrow 'a \ clause \rangle$

assumes

asn-not-empty: $\langle asn \ C \neq \{\} \rangle$ **and**

fml-entails-C: $\langle a \in asn \ C \implies \{map-sign \ fml \ a\} \models_{\mathcal{G}e\sim} \{C\} \rangle$ **and**

C-entails-fml: $\langle a \in asn \ C \implies \{C\} \models_{\mathcal{G}e\sim} \{map-sign \ fml \ a\} \rangle$

begin

interpretation *entails- $\mathcal{G}$ -disj-sound-inf-system*:

Calculi-And-Annotations.sound-inference-system $F\text{-Inf}$ $\langle\{\#\}\rangle$ $\langle(\models_{\mathcal{G}}e)\rangle$
 $\langle\text{proof}\rangle$

interpretation *LA-is-calculus*: *calculus* $\langle\{\#\}\rangle$ $F\text{-Inf}$ $\langle(\models_{\mathcal{G}}e)\rangle$ $F.\text{empty-ord.Red-Red-I}$
 $F.\text{Red-F-}\mathcal{G}\text{-empty}$
 $\langle\text{proof}\rangle$

interpretation *LA-is-sound-calculus*: *sound-calculus* $\langle\{\#\}\rangle$ $F\text{-Inf}$ $\langle(\models_{\mathcal{G}}e)\rangle$ $\langle(\models_{\mathcal{G}}e)\rangle$
 $F.\text{empty-ord.Red-Red-I}$ $F.\text{Red-F-}\mathcal{G}\text{-empty}$
 $\langle\text{proof}\rangle$

interpretation *LA-is-AF-calculus*: *calculus-with-annotated-consrel* $\langle\{\#\}\rangle$ $F\text{-Inf}$ $\langle(\models_{\mathcal{G}}e)\rangle$ $\langle(\models_{\mathcal{G}}e)\rangle$
 $F.\text{empty-ord.Red-Red-I}$ $F.\text{Red-F-}\mathcal{G}\text{-empty}$ *fml asn*
 $\langle\text{proof}\rangle\langle\text{proof}\rangle$

interpretation *core-LA-calculus*: *splitting-calculus* $\langle\{\#\}\rangle$ $F\text{-Inf}$ $\langle(\models_{\mathcal{G}}e)\rangle$ $\langle(\models_{\mathcal{G}}e)\rangle$
 $F.\text{empty-ord.Red-Red-I}$ $F.\text{Red-F-}\mathcal{G}\text{-empty}$ *fml asn*
 $\langle\text{proof}\rangle$

notation *LA-is-AF-calculus.AF-entails-sound* (**infix** $\models_{\mathcal{G}}e_{AF}$ 50)

notation *LA-is-AF-calculus.AF-entails* (**infix** $\models_{\mathcal{G}}e_{AF}$ 50)

interpretation *AF-calculus-extended* $\langle\text{to-AF } \{\#\}\rangle$
 $\text{core-LA-calculus.core.SInf}$ $\langle(\models_{\mathcal{G}}e_{AF})\rangle$ $\langle(\models_{\mathcal{G}}e_{AF})\rangle$ $\text{core-LA-calculus.core.SRed}_I$
 $\text{core-LA-calculus.core.SRed}_F$ $\{\}$ $\{\}$
 $\langle\text{proof}\rangle$

9.7 Lightweight Avatar

We now augment the earlier calculus into *LA* with the simplification rule **BINSPLIT**.

interpretation *with-BinSplit*: *AF-calculus-with-binsplit* $\langle\text{to-AF } \{\#\}\rangle$ $\text{core-LA-calculus.core.SInf}$
 $\text{LA-is-AF-calculus.AF-entails}$ $\text{LA-is-AF-calculus.AF-entails-sound}$ $\text{core-LA-calculus.core.SRed}_I$
 $\text{core-LA-calculus.core.SRed}_F$ $\langle\{\}\rangle$ $\langle\{\}\rangle$ $\text{core-LA-calculus.bin-splittable}$
 $\langle\text{proof}\rangle$

sublocale *LA*: *AF-calculus-extended* $\langle\text{to-AF } \{\#\}\rangle$
 $\text{core-LA-calculus.core.SInf}$ $\text{LA-is-AF-calculus.AF-entails}$ $\text{LA-is-AF-calculus.AF-entails-sound}$
 $\text{core-LA-calculus.core.SRed}_I$ $\text{core-LA-calculus.core.SRed}_F$ *with-BinSplit.Simps-with-BinSplit*
 $\langle\{\}\rangle$
 $\langle\text{proof}\rangle$

By Theorem $\llbracket \text{annotated-calculus } ?bot \text{ } ?FInf \text{ } ?entails \text{ } ?entails-sound \text{ } ?FRed_I \text{ } ?FRed_F \text{ } ?fml \text{ } ?asn; \text{Calculi-And-Annotations.statically-complete-calculus } ?bot \text{ } ?FInf \text{ } ?entails \text{ } ?FRed_I \text{ } ?FRed_F \rrbracket \implies \text{Calculi-And-Annotations.statically-complete-calculus } (to-AF \text{ } ?bot) \text{ } (\text{annotated-calculus}.SInf \text{ } ?bot \text{ } ?FInf) \text{ } (\text{calculus-with-annotated-consrel}.AF\text{-entails } ?entails) \text{ } (\text{annotated-calculus}.SRed_I \text{ } ?bot \text{ } ?FInf \text{ } ?FRed_I) \text{ } (\text{annotated-calculus}.SRed_F \text{ } ?FRed_F)$, we can show that LA is statically complete, and therefore dynamically complete by Theorem $\llbracket \text{annotated-calculus } ?bot \text{ } ?FInf \text{ } ?entails \text{ } ?entails-sound \text{ } ?FRed_I \text{ } ?FRed_F \text{ } ?fml \text{ } ?asn; \text{Calculi-And-Annotations.statically-complete-calculus } ?bot \text{ } ?FInf \text{ } ?entails \text{ } ?FRed_I \text{ } ?FRed_F \rrbracket \implies \text{Calculi-And-Annotations.dynamically-complete-calculus } (to-AF \text{ } ?bot) \text{ } (\text{annotated-calculus}.SInf \text{ } ?bot \text{ } ?FInf) \text{ } (\text{calculus-with-annotated-consrel}.AF\text{-entails } ?entails) \text{ } (\text{annotated-calculus}.SRed_I \text{ } ?bot \text{ } ?FInf \text{ } ?FRed_I) \text{ } (\text{annotated-calculus}.SRed_F \text{ } ?FRed_F)$.

lemma $F\text{-disj-complete}$: $\langle \text{statically-complete-calculus } \{\#\} \text{ } F\text{-Inf } (\models_{\mathcal{UG}e}) \text{ } F.empty\text{-ord}.Red\text{-Red-I } F.Red\text{-F-}\mathcal{G}\text{-empty} \rangle$
 $\langle \text{proof} \rangle$

theorem $strong\text{-static-comp}$:

$\langle LA\text{-is-}AF\text{-calculus}.locally\text{-saturated } \mathcal{N} \implies \mathcal{N} \models_{\mathcal{UG}e_{AF}} \{to\text{-}AF \text{ } \{\#\}\} \implies to\text{-}AF \text{ } \{\#\} \in \mathcal{N} \rangle$
 $\langle \text{proof} \rangle$

sublocale $strong\text{-statically-complete-annotated-calculus } \langle \{\#\} \rangle \text{ } F\text{-Inf } (\models_{\mathcal{UG}e}) \text{ } (| \models_{\mathcal{UG}e})$
 $F.empty\text{-ord}.Red\text{-Red-I } F.Red\text{-F-}\mathcal{G}\text{-empty } fml \text{ } asn \text{ } core\text{-}LA\text{-calculus}.core.SInf$
 $core\text{-}LA\text{-calculus}.core.SRed_I \text{ } core\text{-}LA\text{-calculus}.core.SRed_F$
 $\langle \text{proof} \rangle$

theorem $strong\text{-dynamic-comp}$: $\langle is\text{-derivation } core\text{-}LA\text{-calculus}.core.S\text{-calculus}.derive \text{ } \mathcal{N}i \implies$

$LA\text{-is-}AF\text{-calculus}.locally\text{-fair } \mathcal{N}i \implies llhd \text{ } \mathcal{N}i \models_{\mathcal{UG}e_{AF}} \{to\text{-}AF \text{ } \{\#\}\} \implies$
 $(\exists i. to\text{-}AF \text{ } \{\#\} \in llnth \text{ } \mathcal{N}i \text{ } i) \rangle$
 $\langle \text{proof} \rangle$

sublocale $strong\text{-dynamically-complete-annotated-calculus } \langle \{\#\} \rangle \text{ } F\text{-Inf } (\models_{\mathcal{UG}e}) \text{ } (| \models_{\mathcal{UG}e})$
 $F.empty\text{-ord}.Red\text{-Red-I } F.Red\text{-F-}\mathcal{G}\text{-empty } fml \text{ } asn \text{ } core\text{-}LA\text{-calculus}.core.SInf$
 $core\text{-}LA\text{-calculus}.core.SRed_I \text{ } core\text{-}LA\text{-calculus}.core.SRed_F$
 $\langle \text{proof} \rangle$

end

end

References

- [1] G. Bergeron, F. Krasnopol, and S. Tourret. Formalizing splitting in isabelle/hol. In *ITP (to appear in)*, volume ?? of *LIPICs*, page ?? Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.
- [2] G. Ebner, J. Blanchette, and S. Tourret. A unifying splitting framework. In *CADe*, volume 12699 of *Lecture Notes in Computer Science*, pages 344–360. Springer, 2021.
- [3] G. Ebner, J. Blanchette, and S. Tourret. Unifying splitting. *J. Autom. Reason.*, 67(2):16, 2023.